



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1971

Hybrid sonar simulation.

Pereira, Mauro Cesar Rodriques.

Monterey, California ; Naval Postgraduate School

<http://hdl.handle.net/10945/15745>

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

HYBRID SONAR SIMULATION

Mauro Cesar Rodrigues Pereira

United States Naval Postgraduate School



THE SIS

HYBRID SONAR SIMULATION

by

Mauro Cesar Rodrigues Pereira

Thesis Advisor:

G. A. Rahe

September 1971

Approved for public release; distribution unlimited.

Hybrid Sonar Simulation

by

Mauro Cesar Rodrigues Pereira
Commander, Brazilian Navy

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
September 1971

Thesis
P3348
C.1

ABSTRACT

The purpose of this study was the development of a computer system for research and instruction in signal processing for underwater sound applications.

The nature of the problem demanded the use of three computers interfaced by hardware and software to operate as a single system. A general purpose analog computer was employed to generate signals and noise, provide analog processing and filtering and to allow the input of actual signals collected in the field. A general purpose digital computer provided for the examination of existing processing methods and for the investigation of new techniques. A high level graphics computer was used for data display and as the system control station which provided for a high degree of human operator processing and interaction.

All the hardware required was provided in the Computer Laboratory of the Naval Postgraduate School, Department of Electrical Engineering and this study prepared the software for its convenient and efficient use for this application.

TABLE OF CONTENTS

I.	INTRODUCTION -----	6
A.	PURPOSE -----	6
B.	SYSTEM REQUIREMENTS -----	6
C.	ELEMENTS AVAILABLE -----	7
D.	GENERAL APPROACH TO THE SOLUTION -----	9
E.	ELEMENTS CREATED -----	12
	1. Control -----	12
	2. Signal Generation -----	12
	3. Noise Generation -----	14
	4. Acquisition -----	14
	5. Spectral Analysis -----	15
	6. Processing -----	15
	7. Performance Determination -----	16
	8. Display -----	16
	9. Copying -----	21
F.	SYSTEM CAPABILITIES -----	21
G.	DETAILED DESCRIPTION -----	43
II.	DIGITAL PROCESSING -----	44
A.	OVERLAY STRUCTURE -----	44
B.	MAIN PROGRAM -----	46
C.	SUBROUTINES SEGSAMP, XEGSAMP AND SAMPLE -----	49
D.	SUBROUTINES DISPLAY AND TISPLAY -----	50
E.	SUBROUTINE FASTCOR -----	52
F.	SUBROUTINES ANALYSIS AND SEGFOUR -----	64
G.	SUBROUTINES PROCESS AND PISPLAY -----	68
H.	SUBROUTINE CIRCPAT -----	72
I.	AUXILIARY SUBROUTINES -----	73

J. SUBROUTINES DECO AND CONST -----	74
III. ANALOG CIRCUITS -----	76
A. PATTERN GENERATOR -----	76
1. Theory -----	76
2. Implementation -----	78
3. Logic -----	80
4. Scaling -----	80
5. Operation -----	82
B. HARMONIC GENERATOR -----	83
1. Sine Generators -----	83
2. Multipliers -----	83
C. WHITE NOISE GENERATOR -----	85
1. Theory -----	85
2. Implementation -----	86
D. SIGNAL COMBINER -----	88
E. BANDPASS FILTER -----	88
1. Narrow-band Filter -----	88
2. Lowpass Filter -----	91
F. INTERRUPT GENERATOR -----	92
G. AUXILIARY CIRCUITS -----	94
IV. GRAPHICS TERMINAL -----	98
V. SONAR SYSTEM APPLICATIONS -----	99
A. PRELIMINARY CONSIDERATIONS -----	99
B. PERFORMANCE COMPARISON -----	101
C. INTEGRATION TIME X BANDWIDTH -----	102
D. INFORMATION BANDWIDTH -----	103
VI. CONCLUSION AND RECOMMENDATIONS -----	105

APPENDIX A	SYSTEM OPERATION -----	108
APPENDIX B	ANALOG COMPUTER INFORMATION -----	113
COMPUTER PROGRAM	-----	119
BIBLIOGRAPHY	-----	185
INITIAL DISTRIBUTION LIST	-----	186
FORM DD 1473	-----	187

I. INTRODUCTION

A. PURPOSE

The purpose of this work was to develop a convenient method and efficient programs to employ the unique computer system of the Naval Postgraduate School Computer Laboratory to build a hybrid system for signal analysis and processing. The primary good of the system was to achieve a high speed computational tool with a high degree of human operator interaction. In addition the system was to provide the flexibility and interaction which would permit the operator to use his heuristic capabilities in approximating solutions in a design procedure when the search for adequate patterns and processors is the goal, or to permit the student to verify theoretical results for a wide variety of situations that he may configure almost at will.

It was also assumed from the beginning that the system should be directly aimed to the underwater sound detection problem. This decision not only dictated the name of the system namely a Hybrid Sonar Simulation, but also directed the choice of the preprogramed applications initially included. This choice does not, however, restrict the application of the program to a much wider field of signal processing limited primarily only by the user's imagination and expertise.

B. SYSTEM REQUIREMENTS

After establishing the general purpose, some specifications were set up to further specify the scope to be achieved.

In order to enable the user an easy and rapid perception of the signals and the results of their analysis or processing, a graphical

display was chosen. To allow him to interact with the process, it was necessary to make available means of controlling the signals, processors and operations at the same location where the situation was being displayed and while it was shown. The fundamental requirement for this type of operation is very high speed computation.

Since comparison of signals and patterns was one of the main objectives, the user should be able to choose among different classes of signals and then vary their parameters. Also the characteristics of filters should be under control.

For analysis and processing there was established the requirement for spectral analysis, auto- and cross-correlation and convolution.

Three types of processors, a linear cross-correlator and two non-linear processors were included as well as the capabilities to determine signal-to-noise ratio, input-output performance characteristics ("processing gain") and the receiver operating characteristics (ROC).

Random noise was necessary to be mixed with the uncorrupted signals in controlled amounts. A Gaussian-like noise was considered adequate for initial test purposes.

Finally a possibility for taking hard copies of the curves displayed had to be incorporated to allow record keeping for further studies and comparisons.

C. ELEMENTS AVAILABLE

The elements available in the computer laboratory and used to compose the system include (see Fig. 1.1):

Digital computer — XDS 9300 — a medium sized general purpose computer with 32K words of main memory and a processor capable of handling a large set of operations with very good efficiency. In the installation it is

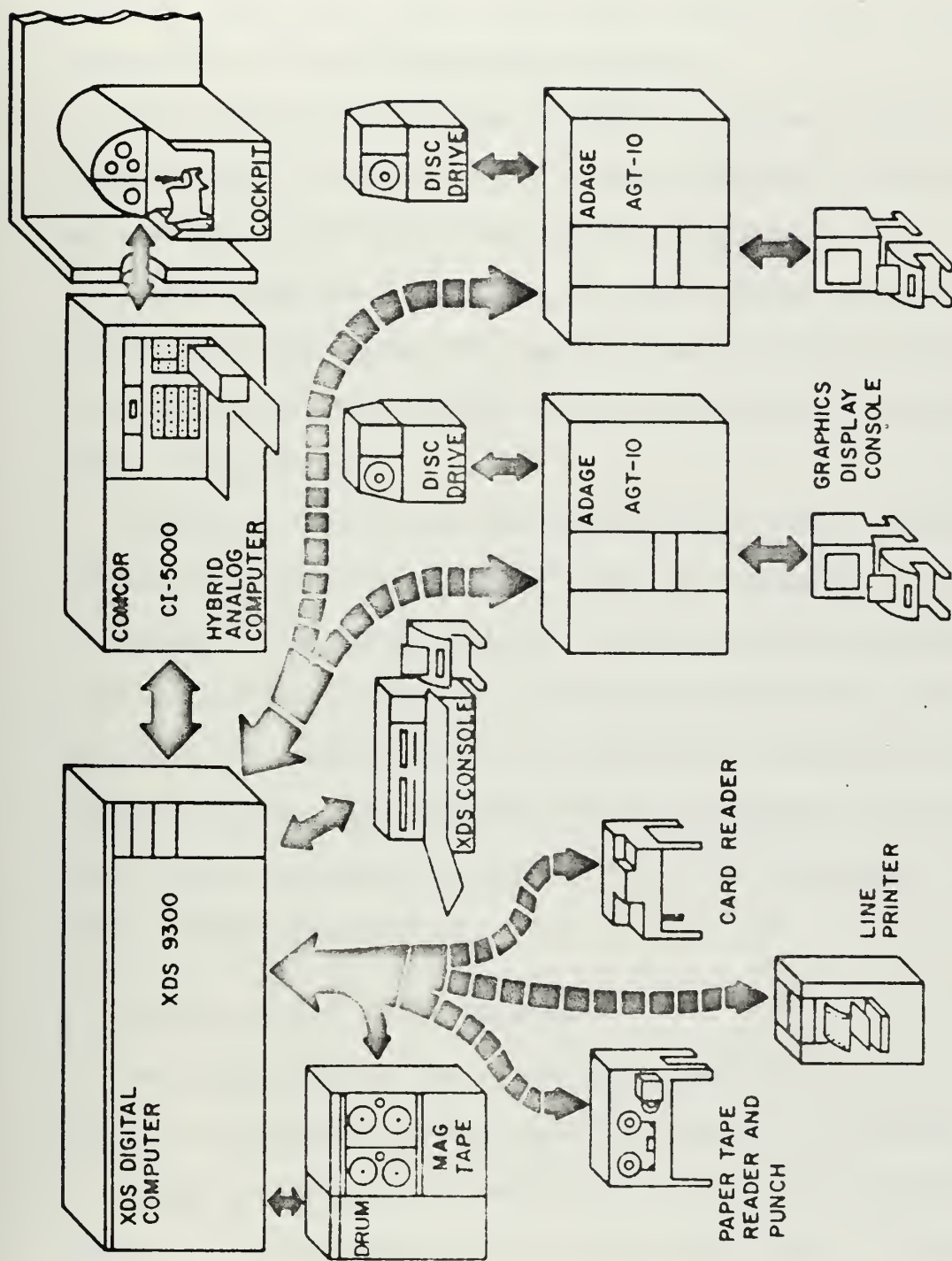


FIGURE I.1 ELECTRICAL ENGINEERING COMPUTER LAB.

connected to the analog computer and the graphics digital computer which allows the hybrid configuration. Also available are several of the standard input-output devices among them a magnetic drum secondary storage with a two million character capacity.

Analog computer — The Comcor CI 5000 is a very well equipped analog computer which includes a parallel logic portion that allows extensive decision making and control functions to be performed.

Graphics computer — The AGT-10 is a small general purpose computer, with 8K words main memory which has as a special feature a CRT output and the necessary hardware to write on the screen, generate vectors and refresh the pictures.

Converters — Both a high precision analog to digital converter, or sampler, and digital to analog converters are available.

Software — The system software includes the normal mathematical functions and subroutines to perform some FORTRAN operations, a set of programs to control the analog computer from the digital, programs to control the graphics terminal, two subroutines to plot curves with the line printer and a program to compute fast Fourier transforms according to the so called Cooley-Tuckey method [Ref. 2].

D. GENERAL APPROACH TO THE SOLUTION

The logical approach was to use the elements available according to their best performance characteristics as applied to the problem.

Naturally, the graphics computer was taken as the data presentation component and then became also the system control unit. Sitting in front of it the engineer incorporates himself into the system as the center of decisions. He selects an operation, analyzes the results, determines

the next step, compares the new results, and introduces new parameters until satisfied with the information accumulated. Since the graphics terminal is itself a general purpose computer, the types of data to be displayed as well as the format of the display may be varied to investigate the effect on Human Operator performance.

In nature the great majority of data or information comes in a continuous or analog form, the discretization being an human abstraction or simplification for easier handling. In particular, that is the situation encountered in the detection problem where the signals are continuous. For these reasons the analog computer was utilized to generate the signals. The analog computer offers the possibility of wider range of variations with very simple and easy to perform change of parameters. By using signals of the same nature as the real ones, practically no changes are required if instead of simulated functions true signals are the object of analysis either in the form of a pre-recorded reproduction or as the output of a receiver.

To analyze or process the signals the discretization becomes a necessity since the digital machines are by far more precise and very well adapted to the computations described by fixed algorithms. Hence the choice of the digital computer for this part of the system.

Two ways may be used to show the arrangement. From a functional standpoint the operations and their inter-relationship are the constituent blocks as presented in Fig. 1.2. Depending on the manner these functions are employed, two modes of operation result and may be named ANALYSIS and SIMULATION. In the ANALYSIS mode it was intended that basically one function be performed at a time under operator control thereby giving him an opportunity to verify intermediate results and to proceed accordingly.

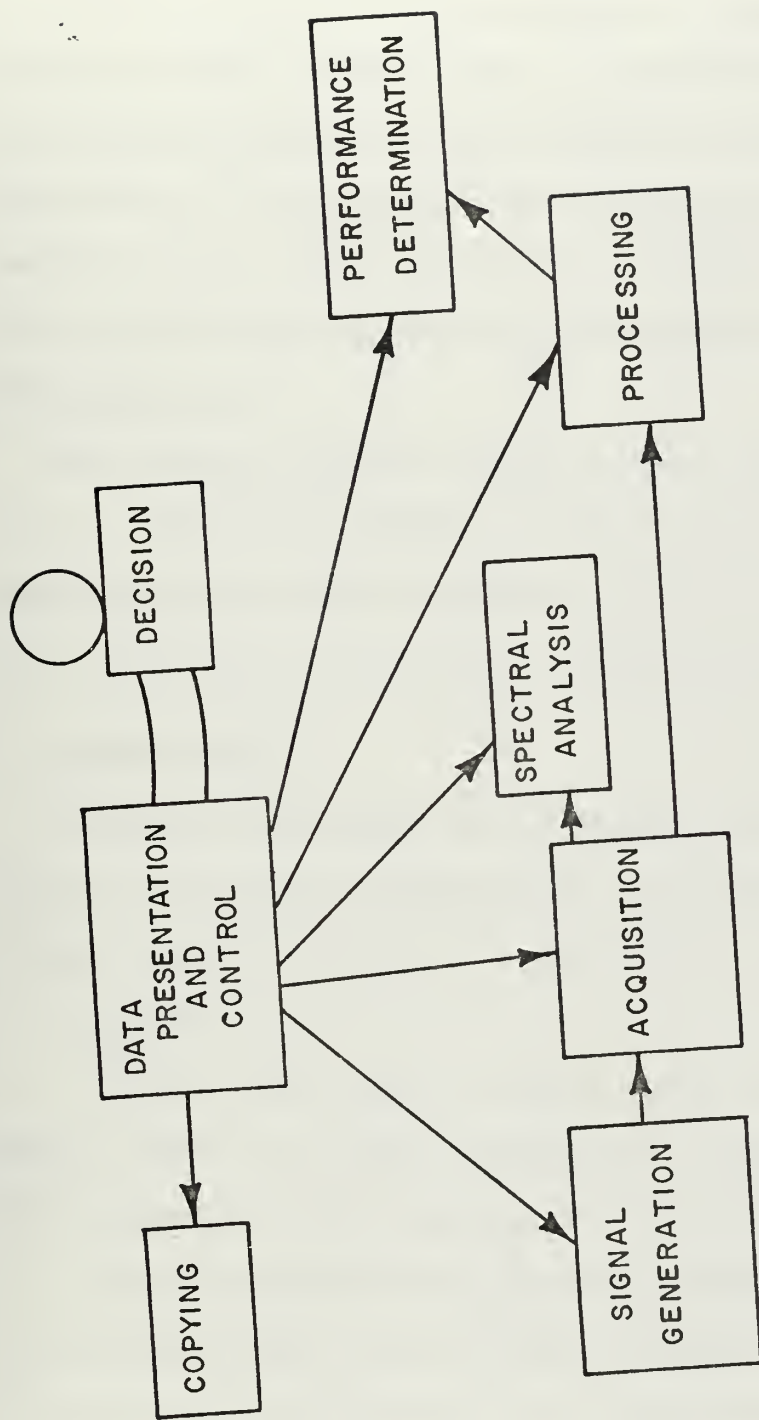


FIGURE 1.2

Through the SIMULATION mode some pre-programmed problems were configured and automatic sequences could be entered to determine processing gain, receiver operating characteristics and detection distance. Some of the processing functions could also be executed by a program that performs correlation by a point by point multiplication and addition; since these operations serve no other purpose than to demonstrate the long time required by this direct approach they were thought of as belonging to a DEMONSTRATION mode.

From a physical point of view the system is described as depicted in Fig. 1.3. The digital computer, besides providing most of the processing functions, exerted effective control of the other units from which it received interrupts to initiate or synchronize operations.

E. ELEMENTS CREATED

To implement this system several elements were created to execute the functions and satisfy the requirements established. A brief description of these elements is presented in the following paragraphs.

1. Control

A program was written to read the control codes typed on the graphics terminal and compare them against a table to establish pointers that indicated the proper sequence to be executed.

Another program was introduced to allow the operator full control of the analog computer from the graphics terminal. Through this feature it is possible to select modes, set or test lines, set potentiometers or scan the various analog components by typing simple messages at the console.

2. Signal Generation

Signal generation was developed from sine generators patched on the analog board. Bearing in mind the sonar application, one of these

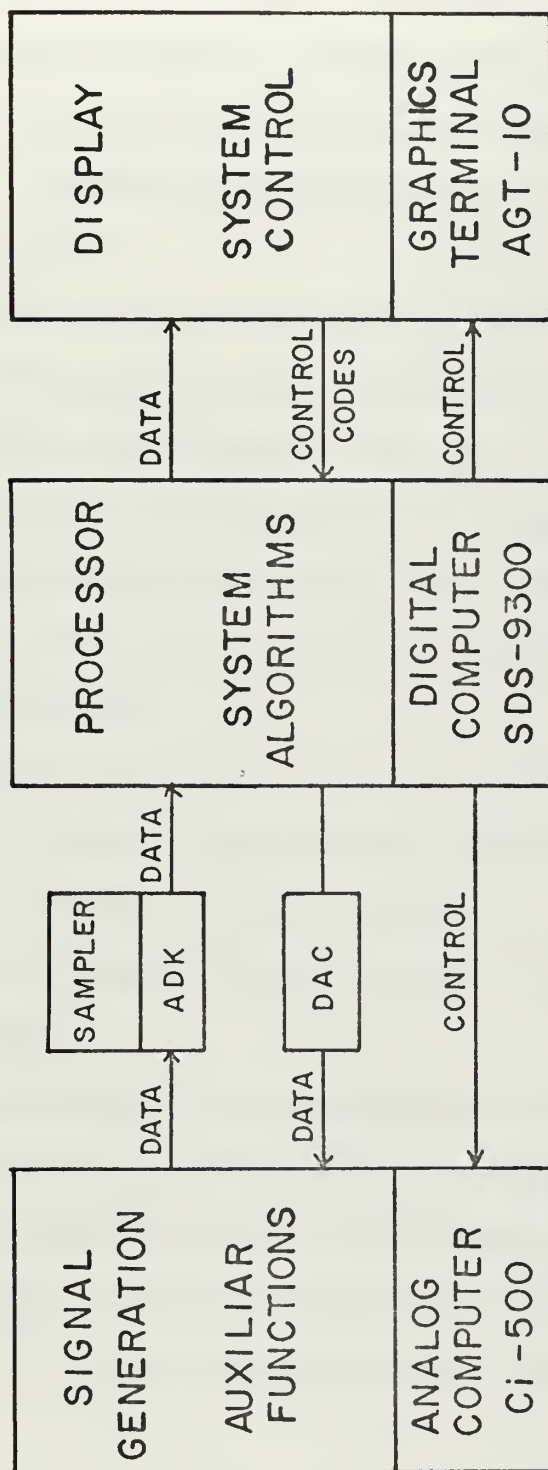


FIGURE 1.3

generators was associated with additional analog and logic circuitry to permit simulation of CW, FM-slide or FSK codes. This generator was also equipped with positive feedback to make possible a more realistic effective sampling rate which required the analog computer to run at a very high frequency. Without some regeneration in this case the wave damping would be intolerable.

Three other sine generators were implemented to simulate propeller signatures, which present a low-pass line spectra composed of a fundamental and several of its harmonics [Ref. 1]. To achieve this result the sine waves were fed to a combination of multipliers whose outputs were summed in controlled amounts to give a final pattern with up to six harmonics, each of which with any desired intensity.

3. Noise Generation

To simulate a random-like noise a pseudo-random binary sequence was generated by a maximal length feedback-shift register and then transformed into a pseudo-random analog square wave. This signal, after proper filtering presents Gaussian statistics at least up to a first-order [Ref. 6].

4. Acquisition

Signal acquisition to the processor was normally made after filtering and then through the sampler or Analog-to-Digital converter (ADK).

Two filters were included: an RC lowpass and a narrowband, high-Q, second-order filter. The latter offered the advantage of being easily controlled from the graphics terminal to adjust frequency, bandwidth or gain.

The purpose of the filters was to represent transducer and part of preprocessor characteristics, but they could also simulate or be preceded by other filters with any other objective, such as transmission medium characteristics.

The sampler was controlled by a program that not only synchronized the operation but also simulated the remaining parts of a pre-processor for the non-linear cases, and stored the samples, conveniently normalized, into the proper locations of the processor memory.

The samples were distinguished and stored as belonging either to the REPLICA or to the SIGNAL. While the choice of names reflects the application to sonar, for a general case these names serve only to identify independent storage areas where signals could be stored for further operation or retrieval. Similarly, a third area where the results of correlations were placed, received the name of CORR.

5. Spectral Analysis

A special subroutine was prepared to make use of the Fast Fourier transform program and prepare the results for display.

6. Processing

A replica correlator simulation was employed. Depending on how the signals have been preprocessed the processor became a linear, amplitude-limited or a clipper-limited type.

The program written to execute the correlation was based on the properties of discrete Fourier transforms and on the time domain convolution theorem as applied to the discrete case. For cross-correlating the replica against a very long time series, as is the case where processing the incoming signal in a detection problem, a partition technique was adopted [Ref. 4].

The correlation could also be evaluated by multiplying both functions, adding the results, shifting and repeating these operations until the replica had been slid entirely across the signal. The time required for this procedure was incompatible with the real-time specification and the method was left for illustration only.

7. Performance Determination

Under this general classification were considered the input-output signal-to-noise ratio characteristic, the receiver operating characteristic and distance measurement.

A program was prepared to automatically sample the replica and the signal, form the signal either with noise alone or with noise plus transmission modulation code, process the received samples, compute power, determine threshold and threshold crossing, evaluate probability of detection or false-alarm and measure distance, all in the proper sequence according to the case.

Power determination, when necessary, was computed through the application of the Wiener-Kinchin relation and Parseval's theorem.

The threshold was normalized to the peak value of the correlator output when no noise was present.

When using these automatic operations, where only the final results and not the waveforms were of interest, the sampling rate utilized was related to twice the information bandwidth. This allowed processing samples with a more realistic information content. Attempts to display the waveforms at these sampling rates showed that while the signals were completely distorted, they revealed amplitude variations at a rate comparable to the significant bandpass. For signal analysis, when the waveforms were required to be displayed without great distortion, the sampling rate had to be considerably greater than twice the highest frequency present.

8. Display

Programs were prepared to display the functions and results covering all cases treated.

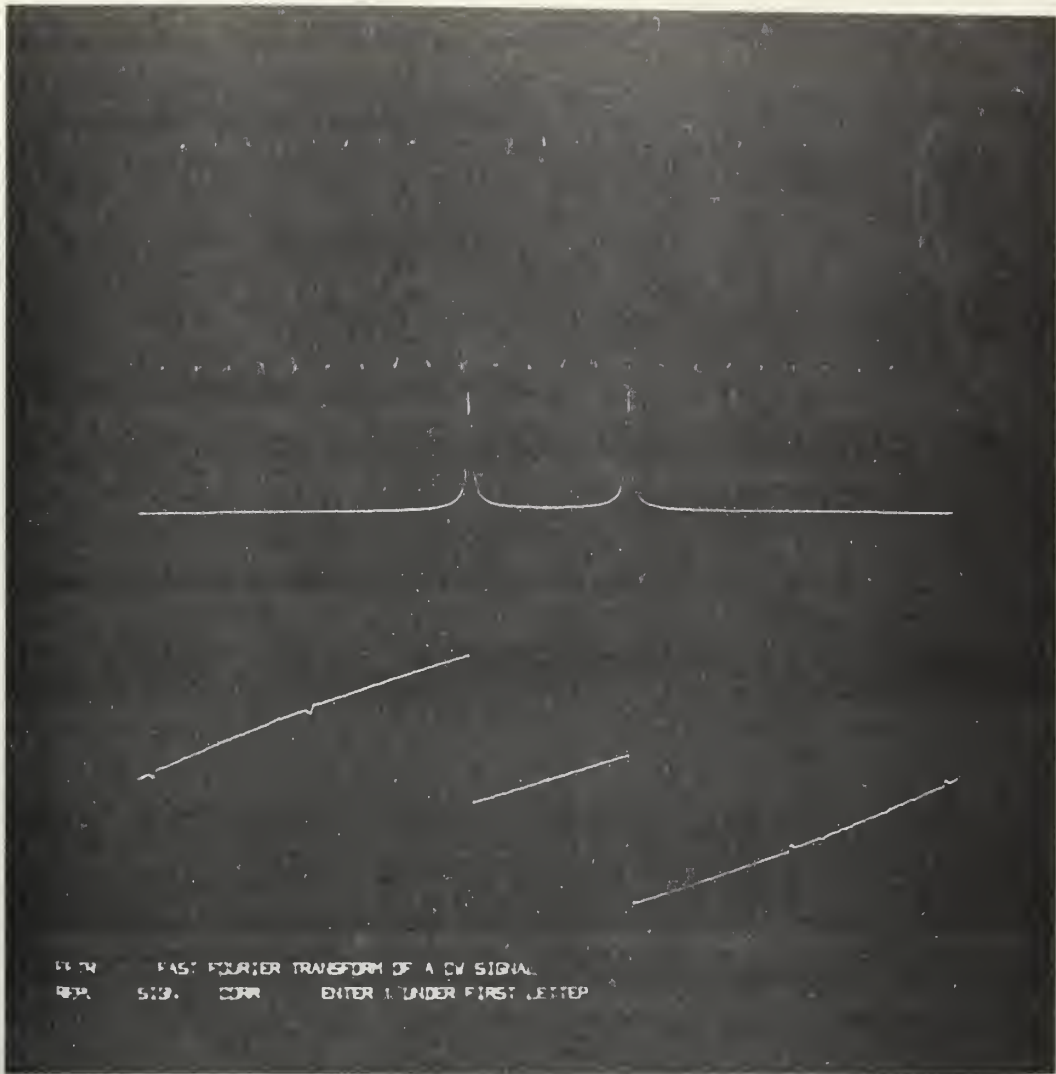


Figure 1.4. Spectrum of a CW signal. This figure is a photograph of the actual display. It shows the time function, absolute value and phase angle of the transform. The location of the impulses, at ± 5000 Hz, serve as reference for the frequency plots in the next pictures.

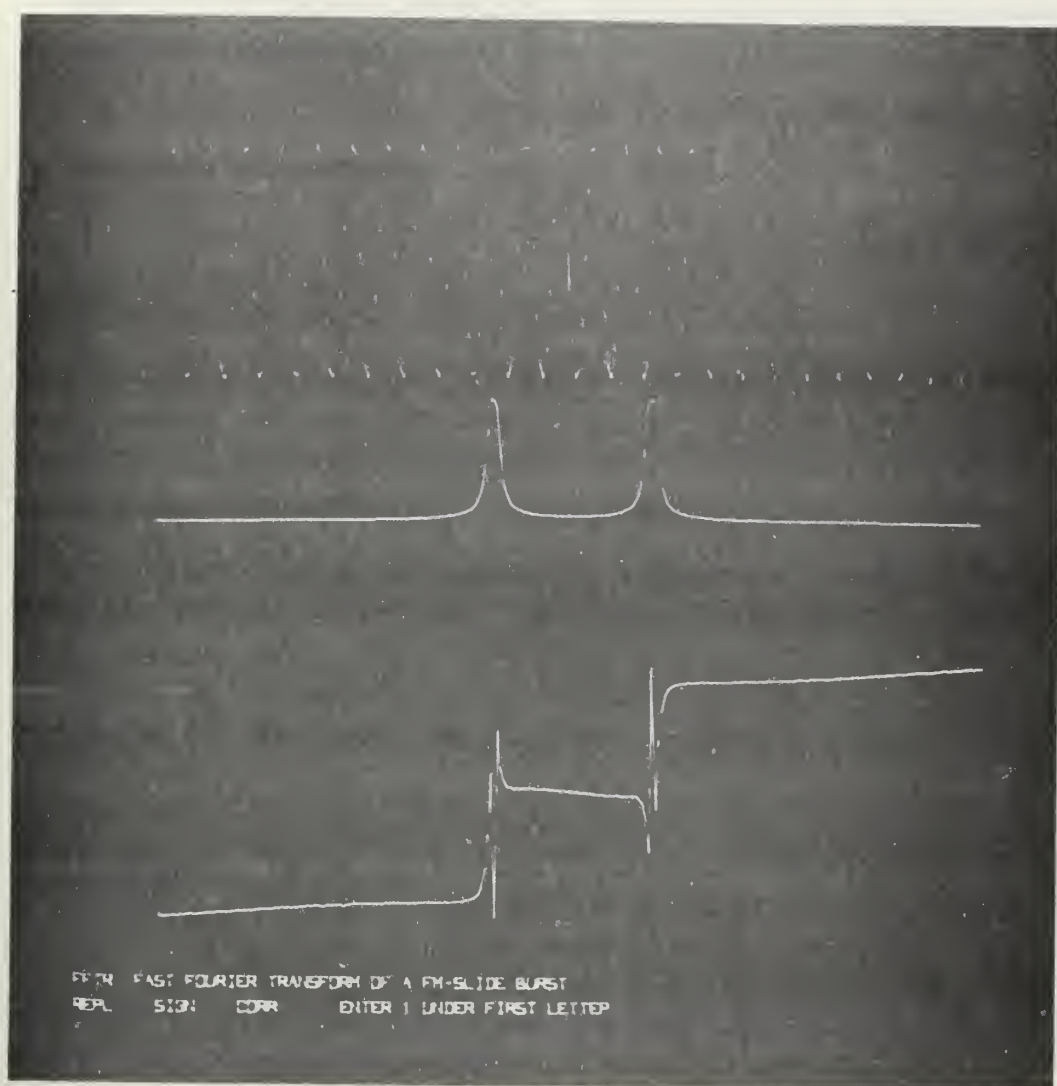


Figure 1.5. Spectrum of an FM signal formed by a 5000 Hz sinusoidal linearly modulated by 1000 Hz. Comparison with Fig. 1.4 shows the wider band covered. Discrepancies observed in the phase angle plot here, as in some of the following pictures, are due to digital computer noise in the computation of arctan.

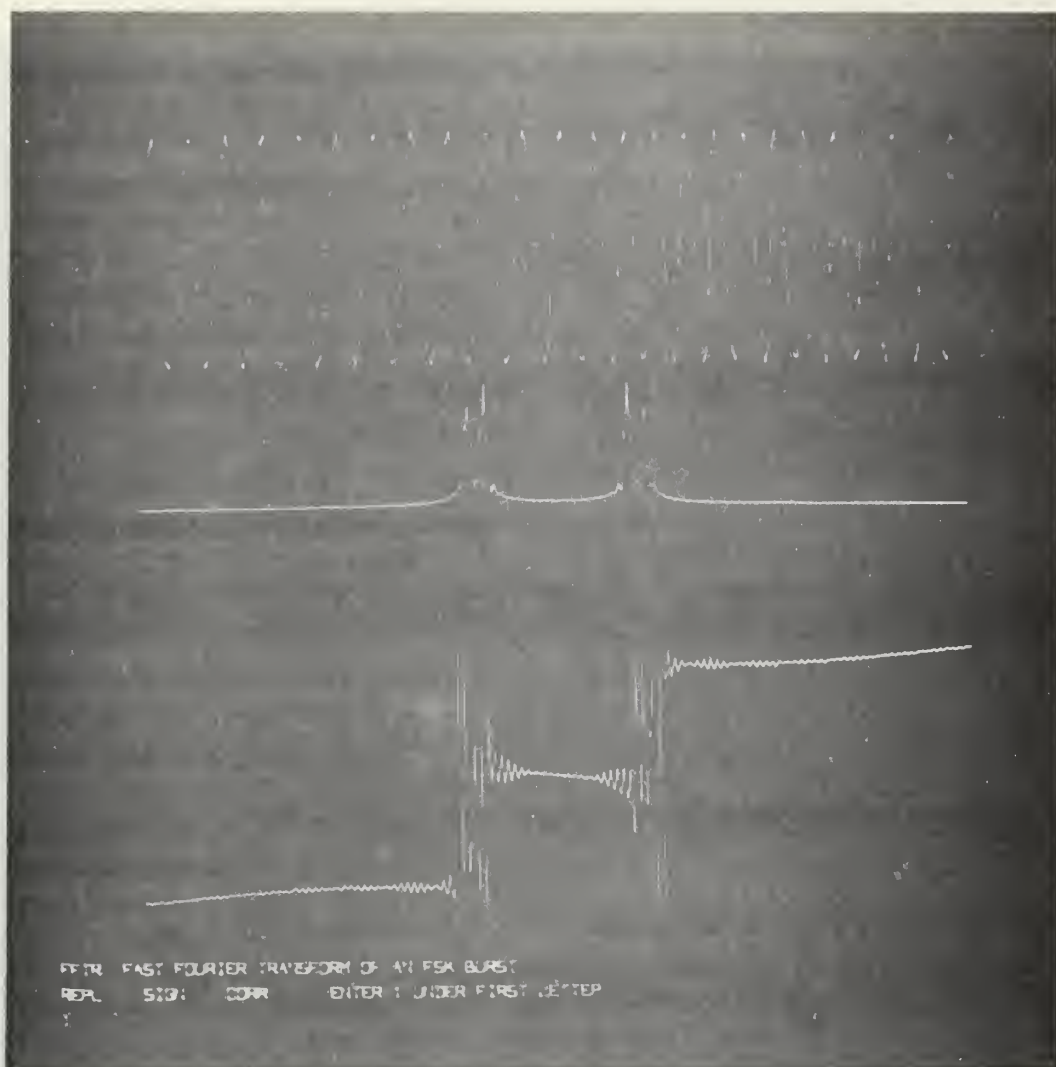


Figure 1.6. Spectrum of a sinusoidal signal of 5000 Hz two-tone frequency modulated by ± 500 Hz.

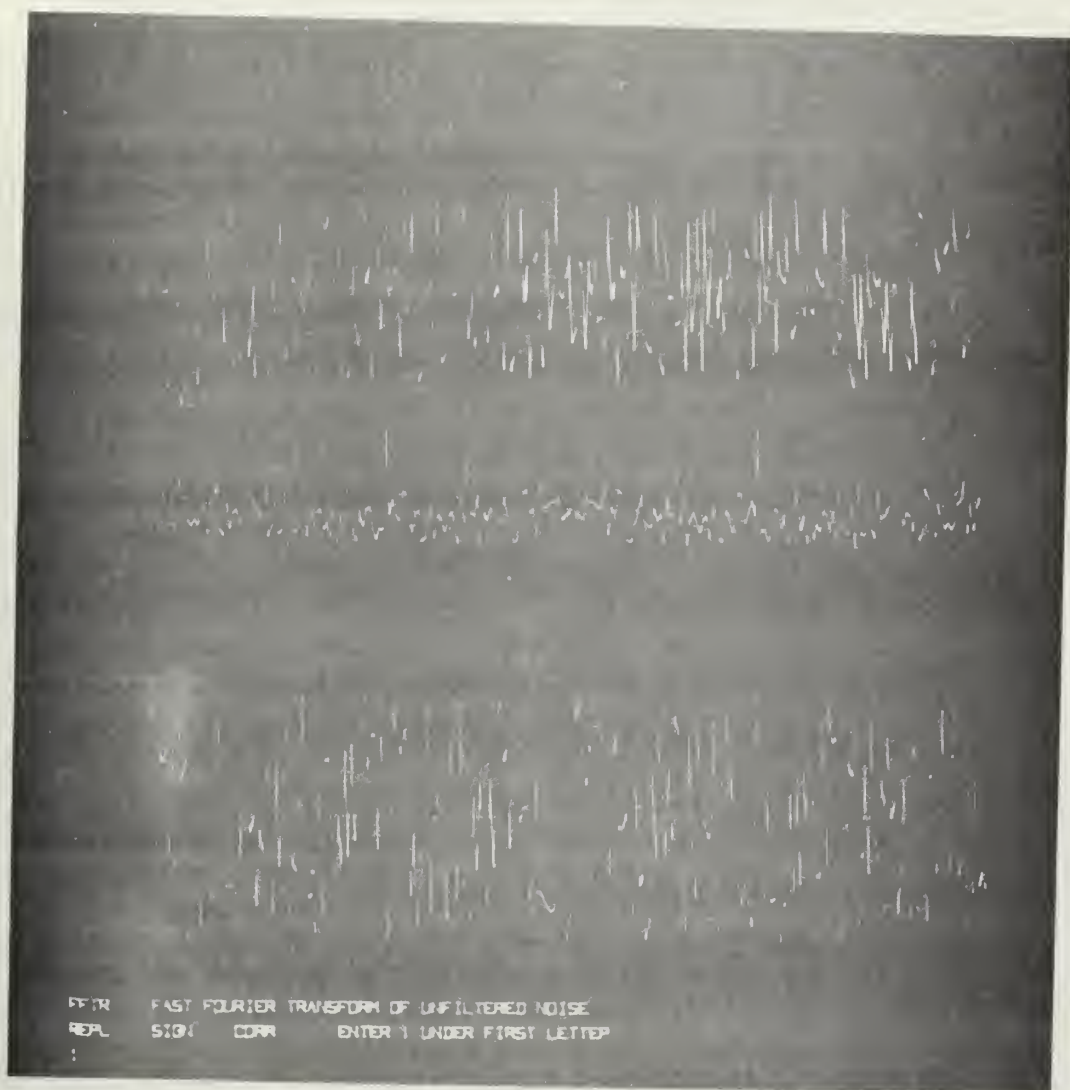


Figure 1.7. Spectrum of unfiltered noise. The unfiltered noise is a pseudo-random square wave which has been passed only through a low-pass filter with cut-off frequency beyond the region of interest. This filtering is necessary to assure Gaussian first-order statistics to the simulated noise. In the band up to 5000 Hz (compare with Fig. 1.4) no significant peaks appear in the spectrum.

9. Copying

A subroutine was written to copy the data displayed on the screen into the format required to use the two programs for line printer plotting.

Also a sequence in this subroutine was prepared to send the information to the analog computer, through the digital to analog converter, in order to plot the curves with a paper recorder. This technique proved very useful in several applications.

F. SYSTEM CAPABILITIES

The system was made flexible enough to allow a wide range of applications. Some of these were checked with very impressive results as demonstrated below.

Initially, Figures 1.4 through 1.8 show types of signals used. Fig. 1.4 presents a CW burst of 5000 Hz and 2.53 msec; also shown is its spectrum, magnitude and phase. In Fig. 1.5 the 5 kHz signal is linearly FM modulated by 1000 Hz and in Fig. 1.6 the 5 kHz carrier has an FSK modulation of ± 500 Hz. Fig. 1.7 depicts a 2.53 msec sample of unfiltered noise; the frequency plot covers the range of ± 50 kHz. Fig. 1.8 is a simulated propeller signature with a fundamental of 60 Hz and six harmonics.

Figure 1.9 shows an example of filter characteristics determination through the application of an impulse at its input. In the upper part the impulse response is presented followed by the magnitude and phase of the transfer function. The filter tested was a narrowband centered at 5 kHz with a bandwidth of less than 1 kHz.

In the next figure, Fig. 1.10, the same random noise previously displayed in Fig. 1.7 is seen but after being passed through the narrow bandpass filter. This experiment was a demonstration that noise behaves as a sinusoidal signal when narrow-band filtered.

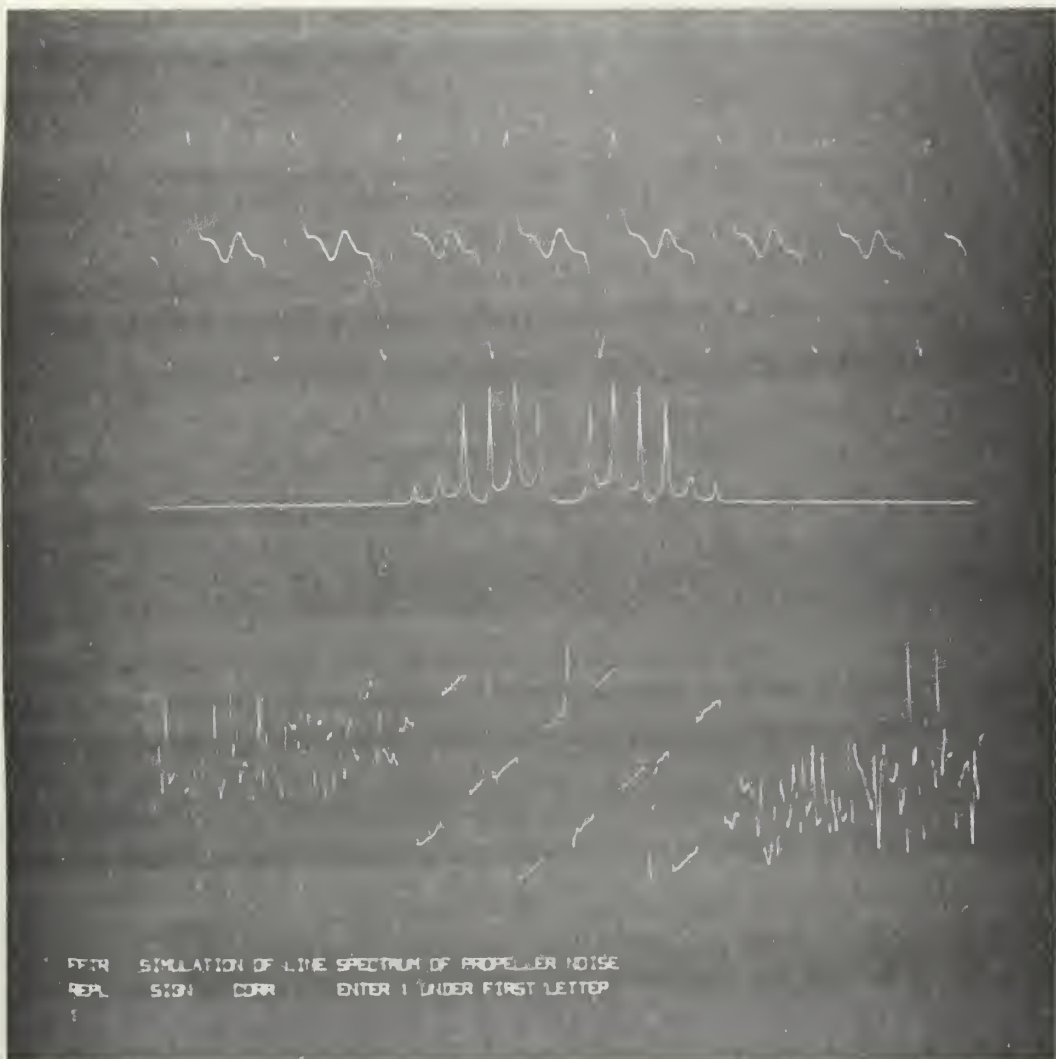


Figure 1.8. Representation of a low-pass periodic signal similar in characteristics to a propeller-generated noise. The line spectrum shows well defined harmonics up to the sixth. The fundamental frequency was 60 Hz.

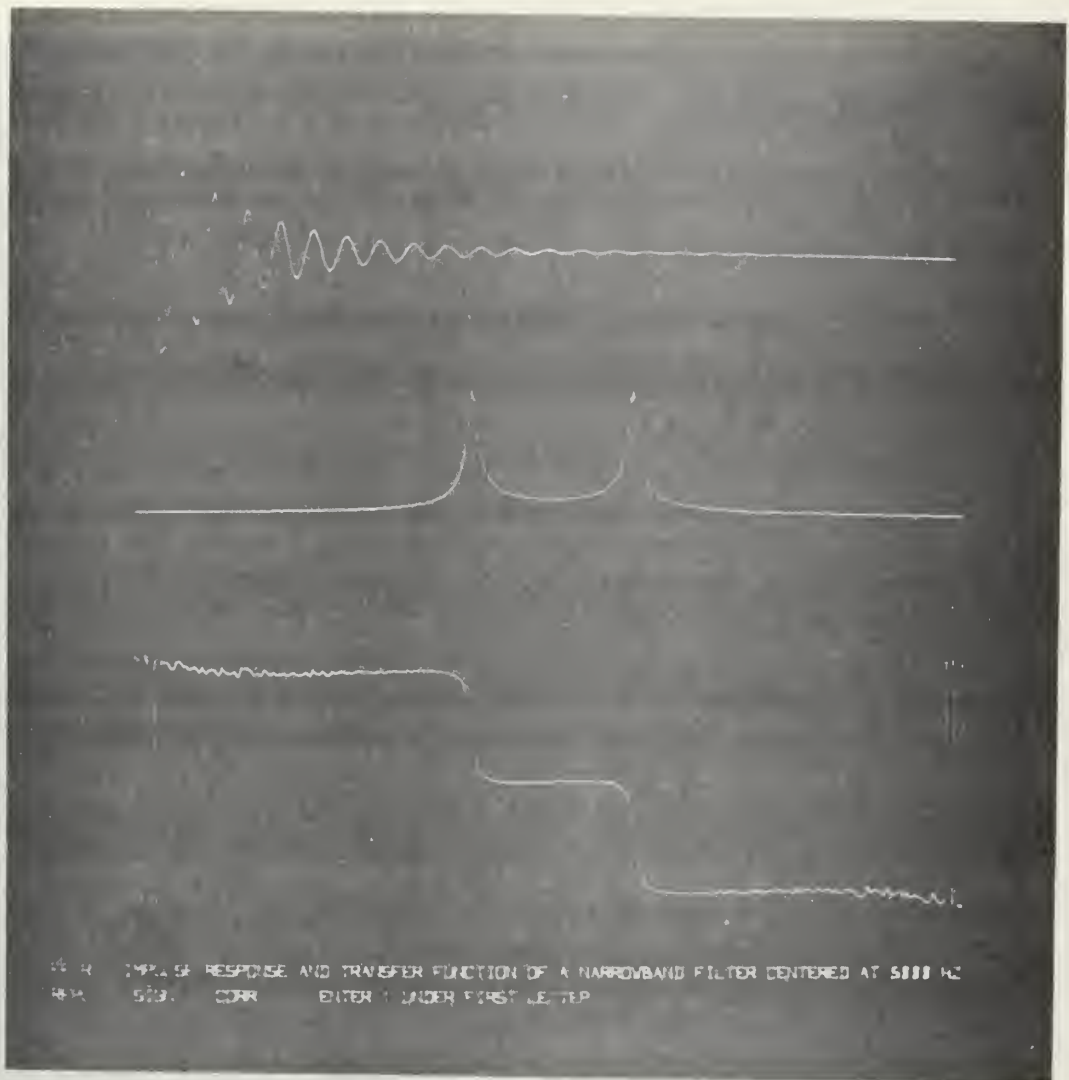


Figure 1.9. This figure presents the narrow-band filter impulse response, and both the amplitude and phase angle of its transform or transfer function.

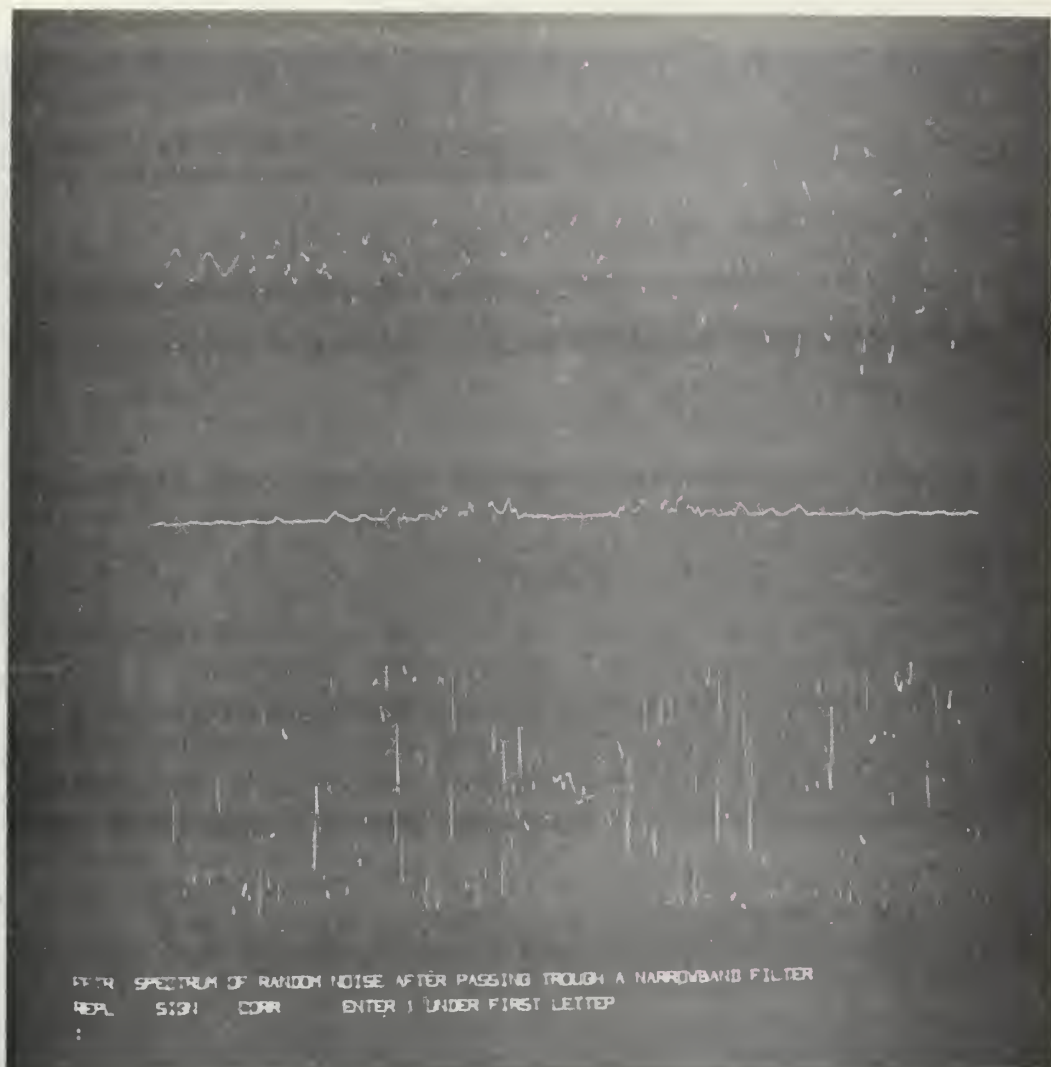


Figure 1.10. In this picture the noise spectrum is seen again but now after narrow-band filtering. The spectral density is similar to that of a sinusoidal signal but the phase angle has a random variation.

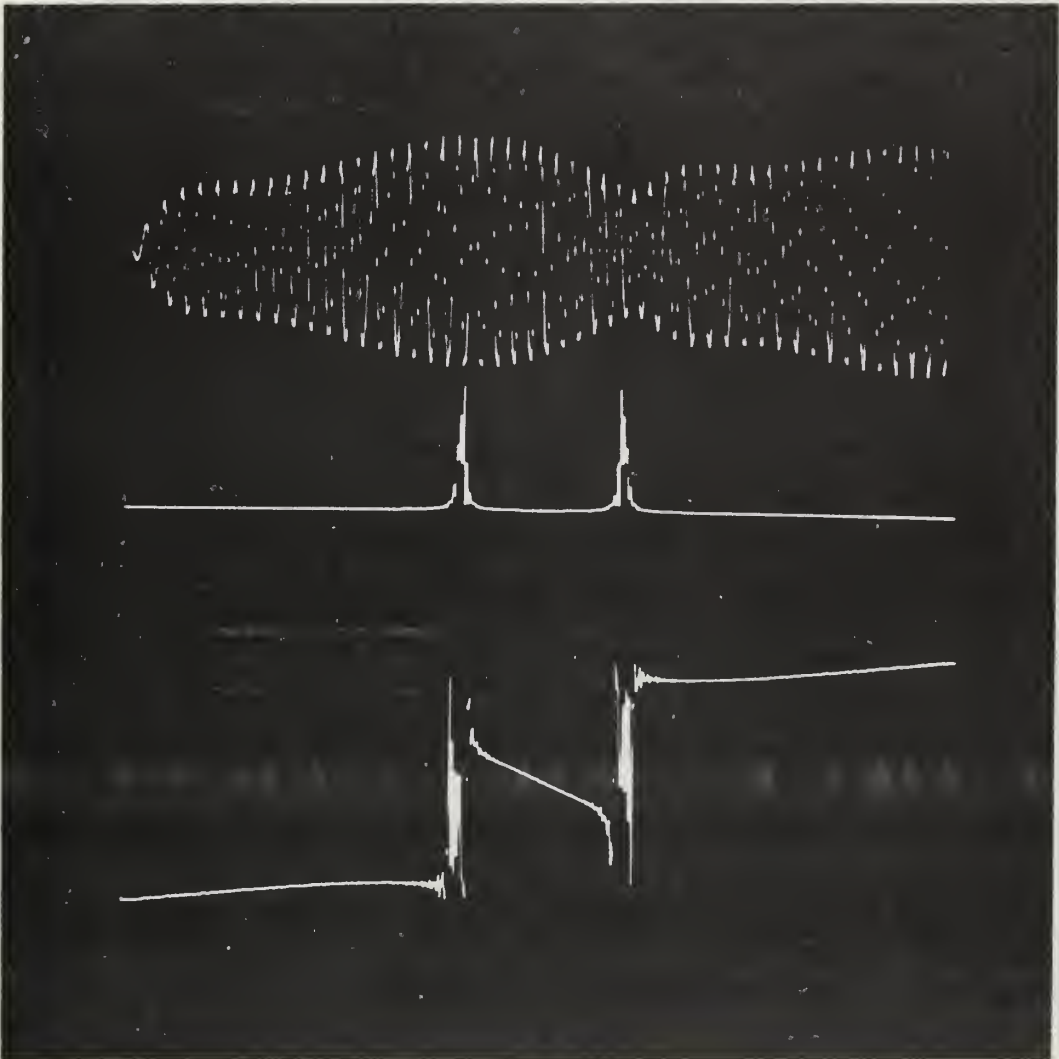


Figure 1.11. The FM-signal of Fig. 1.5 is shown after passed through the narrow-band filter. The effect of the filter narrow-pass is reflected in the time function by a reduction in amplitude for the frequencies above and below the center frequency. In the frequency domain a reduction in bandwidth is observed. The sample in this example is twice as large as that of Fig. 1.5 and shows the effect of the transient filter response.

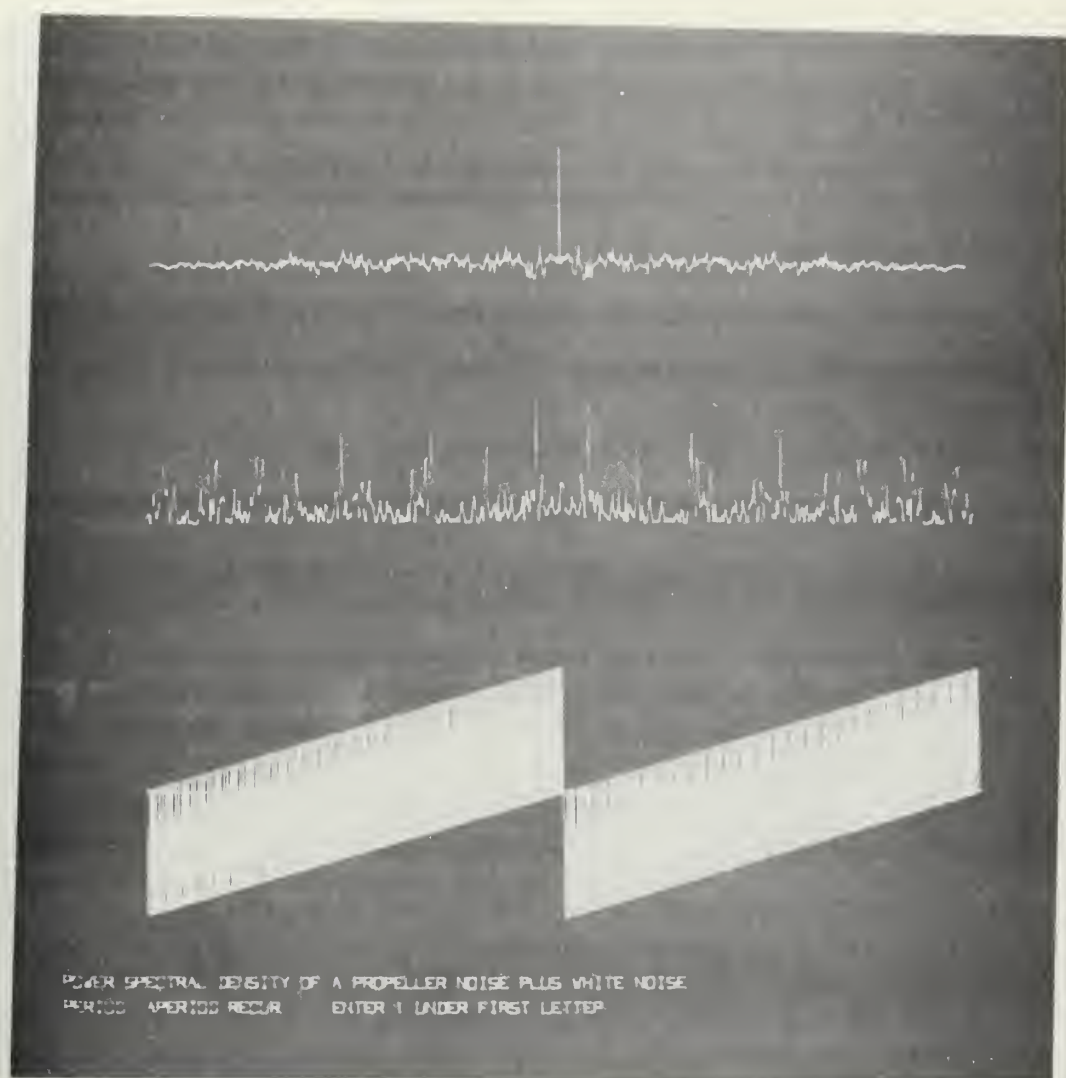


Figure 1.13. Power spectral density of the signal presented in Fig. 1.8 but now highly contaminated by noise. The auto-correlation function and its transform are shown.

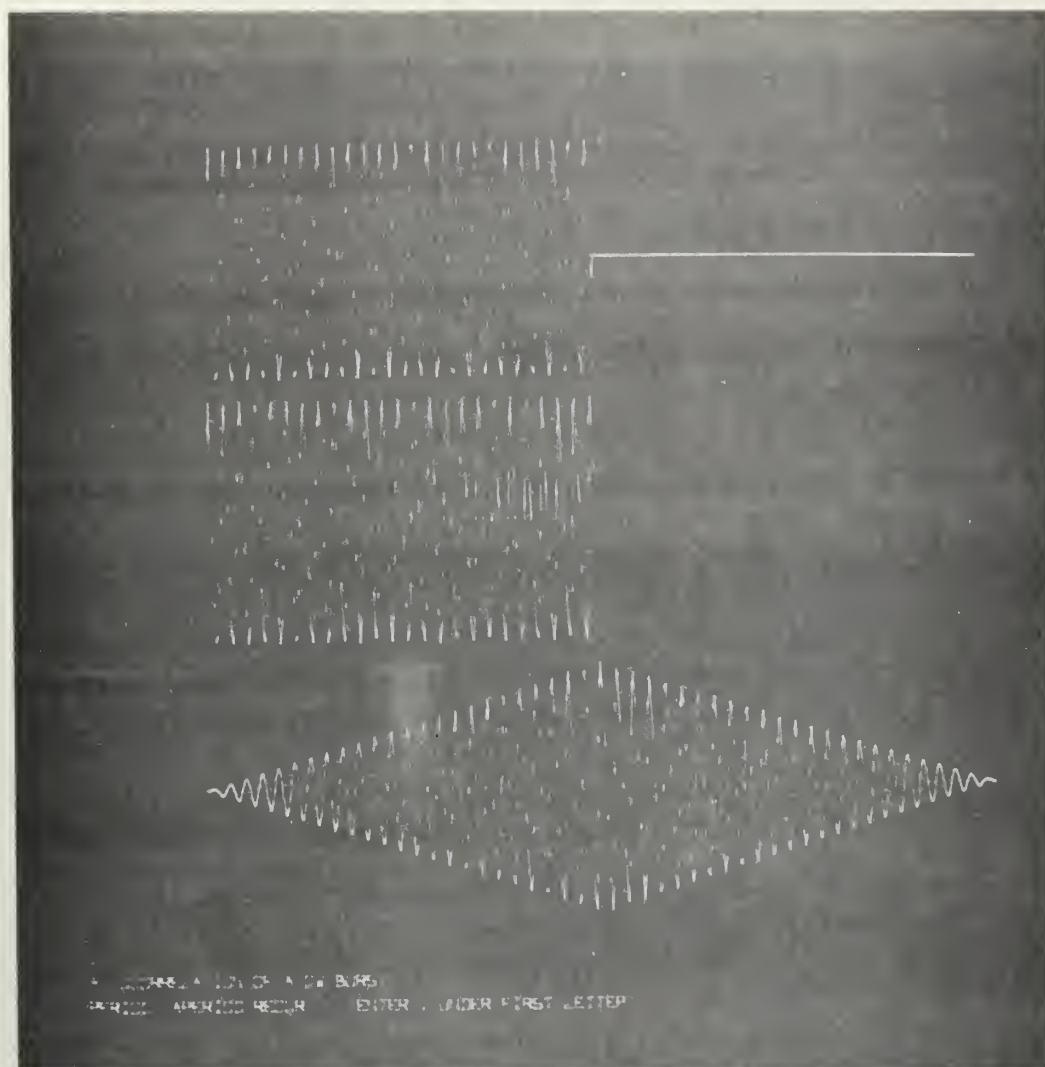


Figure 1.14. Auto-correlation function of a CW-pulse.

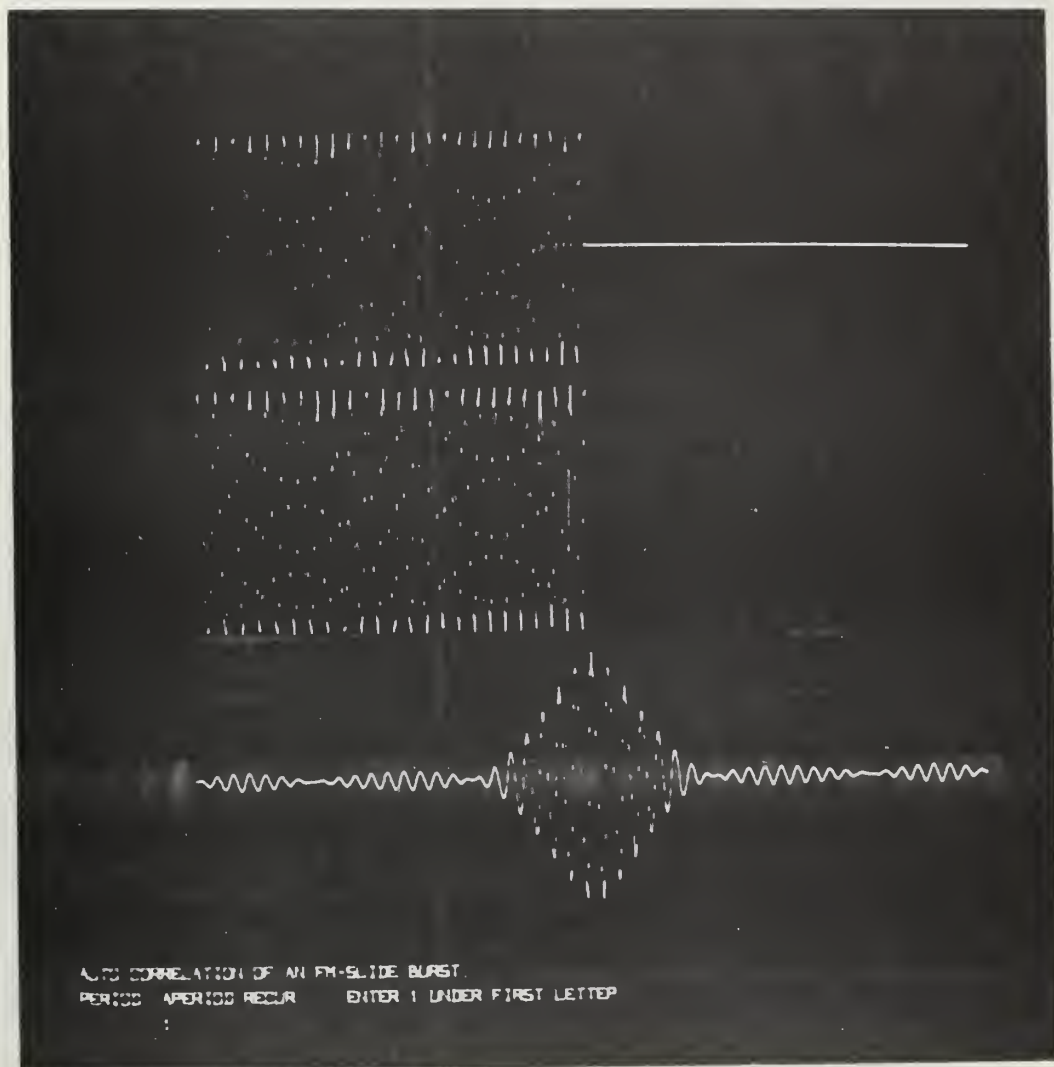


Figure 1.15. Auto-correlation of the FM signal shown previously in Fig. 1.5. Comparison with the auto-correlation of the CW signal presented in Fig. 1.15 shows the better definition of the correlation as a consequence of the wider bandwidth.

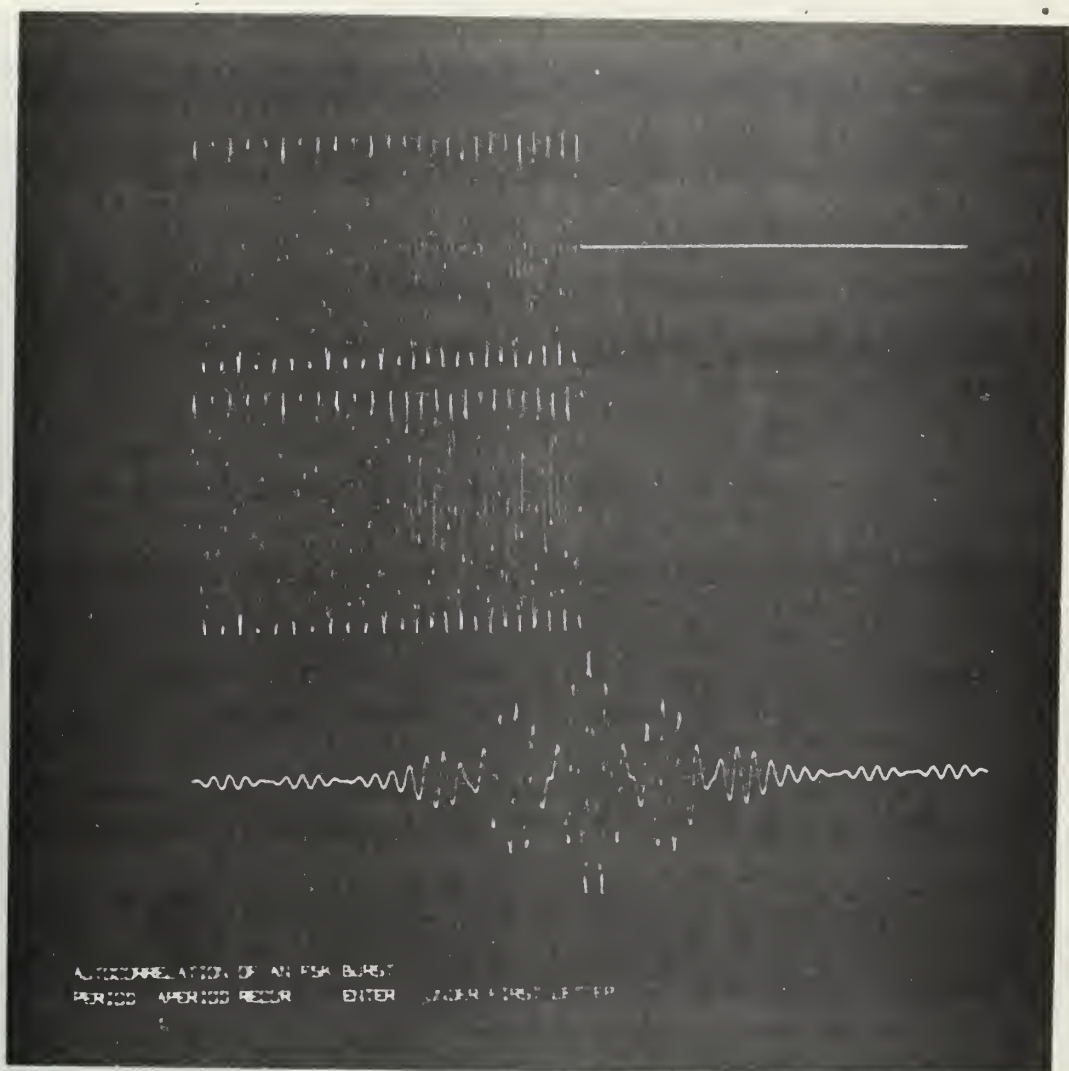


Figure 1.16. Auto-correlation of the FSK pulse presented in Fig. 1.6. By comparison with Fig. 1.15 it is seen that the uneven distribution of the energy over the bandwidth affects the result of the correlation.

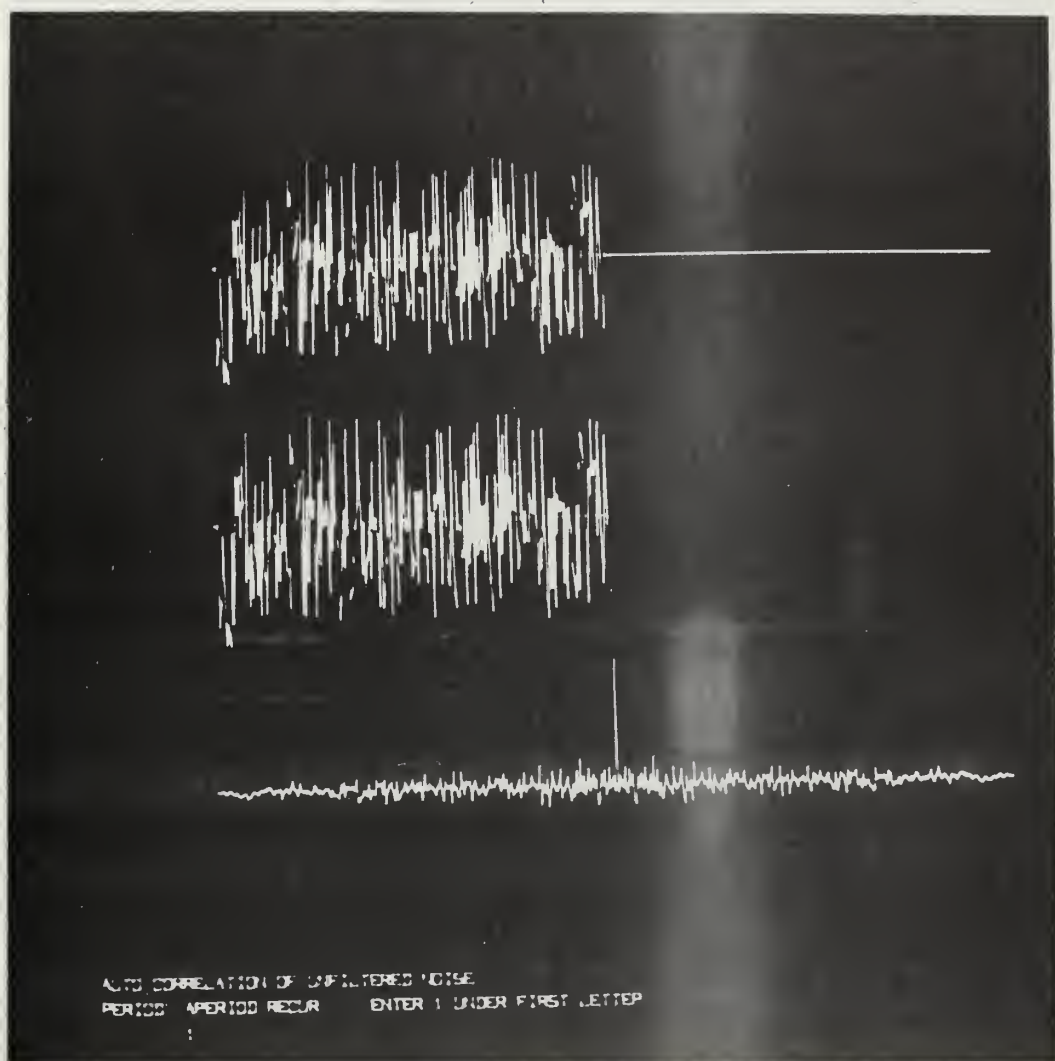


Figure 1.17. Auto-correlation of unfiltered noise. The picture clearly shows the lack of correlation among random-noise samples.

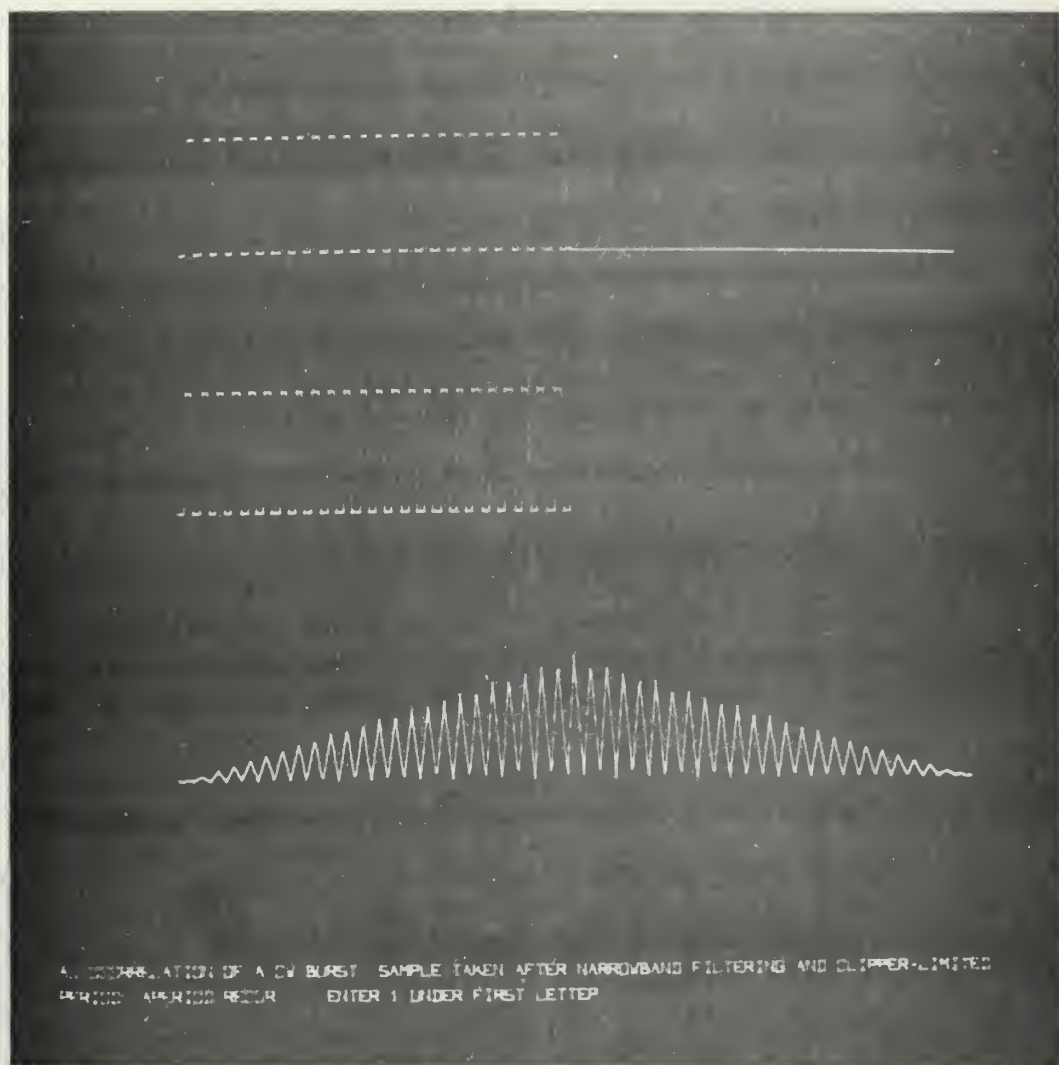


Figure 1.18. In this picture, the CW signal of Fig. 1.4 was amplified and clipped after having been passed through a narrow-band filter. The envelope of the auto-correlation is similar to that of the undistorted signal, which shows that narrow-band signals may be treated by absolute-value processor models.

In Fig. 1.11 the FM signal of Fig. 1.5 is again displayed but now after filtering. In Fig. 1.12 the same result is obtained by convolving the filter impulse response with the unfiltered signal — the upper part of the figure is the filter response flipped, the central the unfiltered signal and the lower the result of convolution. This experiment demonstrated the capability of the system to simulate digital filters or any other filter once the impulse response is known.

Figure 1.13 shows the result of power spectral density evaluation of the propeller signature already mentioned but now noise added and after lowpass filtering. Displayed are the auto-correlation of the signal and the magnitude and phase of the power spectrum.

The next four figures, 1.14, 1.15, 1.16 and 1.17 depict the auto-correlation of the signals previously shown in Figures 1.4 through 1.7, i.e., CW, FM, FSK and noise.

Figure 1.18 shows the auto-correlation of a CW signal processed by a clipper-limited type of processor. Since only the zero crossing information is kept, the resultant signal is a binary signal.

The following figures present the results of the cross-correlation of a replica and a signal composed of the transmission pattern plus noise. The cross-correlation was normalized in such a way that if the transmission code only was present the peak output would be one. All samples are taken after narrowband filtering and correspond to the FM-slide signal.

Figures 1.19 and 1.20 refer to a linear cross-correlator output for two different signal-to-noise ratios while Figs. 1.21 and 1.22 use the same SNR but applied to a clipper-limited processor. The experiment showed that for high signal-to-noise ratio the non-linear processor has a much lower sensitivity than the linear one; the difference, however, is almost non-existent for a low-input signal-to-noise ratio.

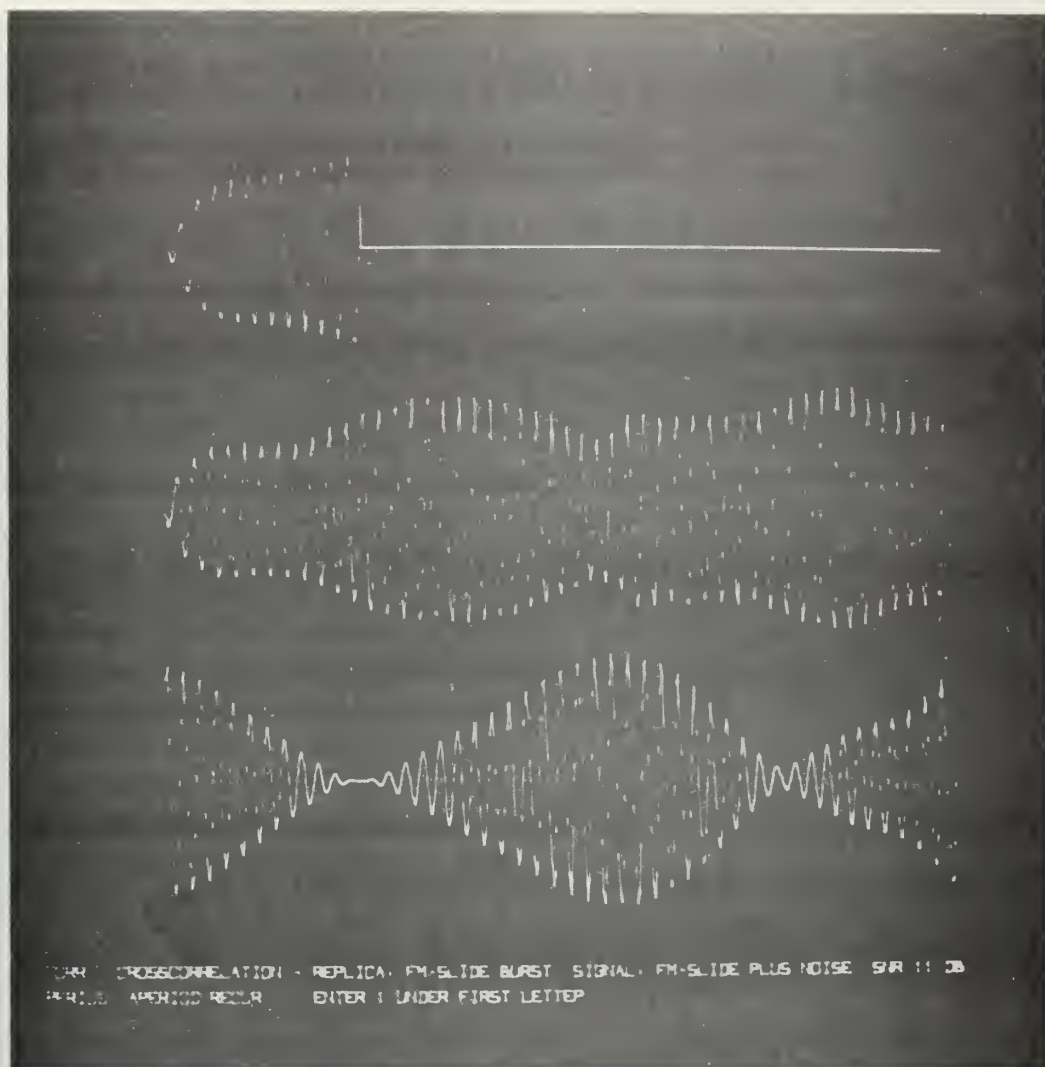


Figure 1.19. Cross-correlation of an FM-replica against this signal plus noise at a high signal-to-noise ratio. The periodic or circular nature of the Discrete Fourier Transform is evident from the result of the correlation.

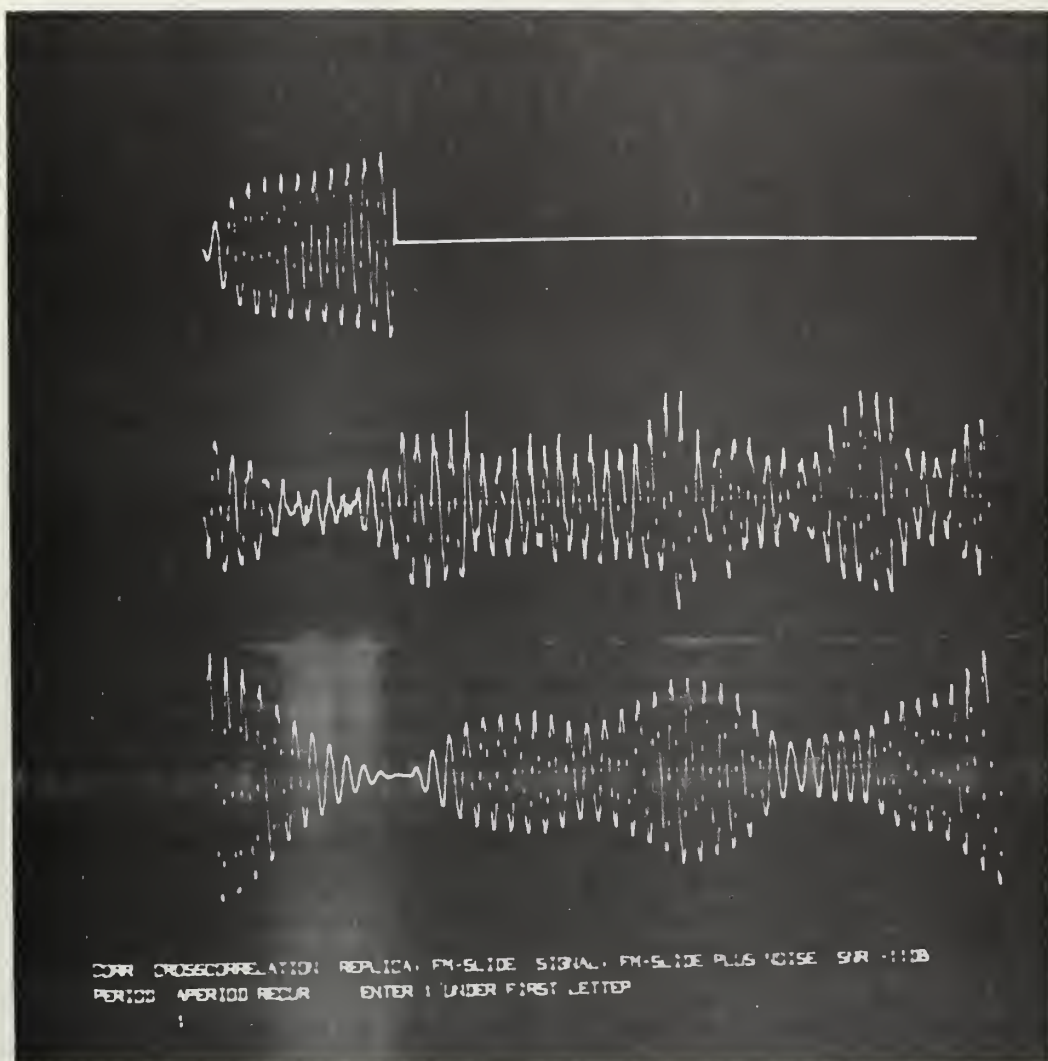


Figure 1.20. This picture is the result of the same operation as that used to obtain Fig. 1.19, but applied when the signal-to-noise ratio is low. Although the functions are normalized to the same maximum amplitude in the display, the deterioration caused by noise is evident, especially when the central "bulge" is observed.

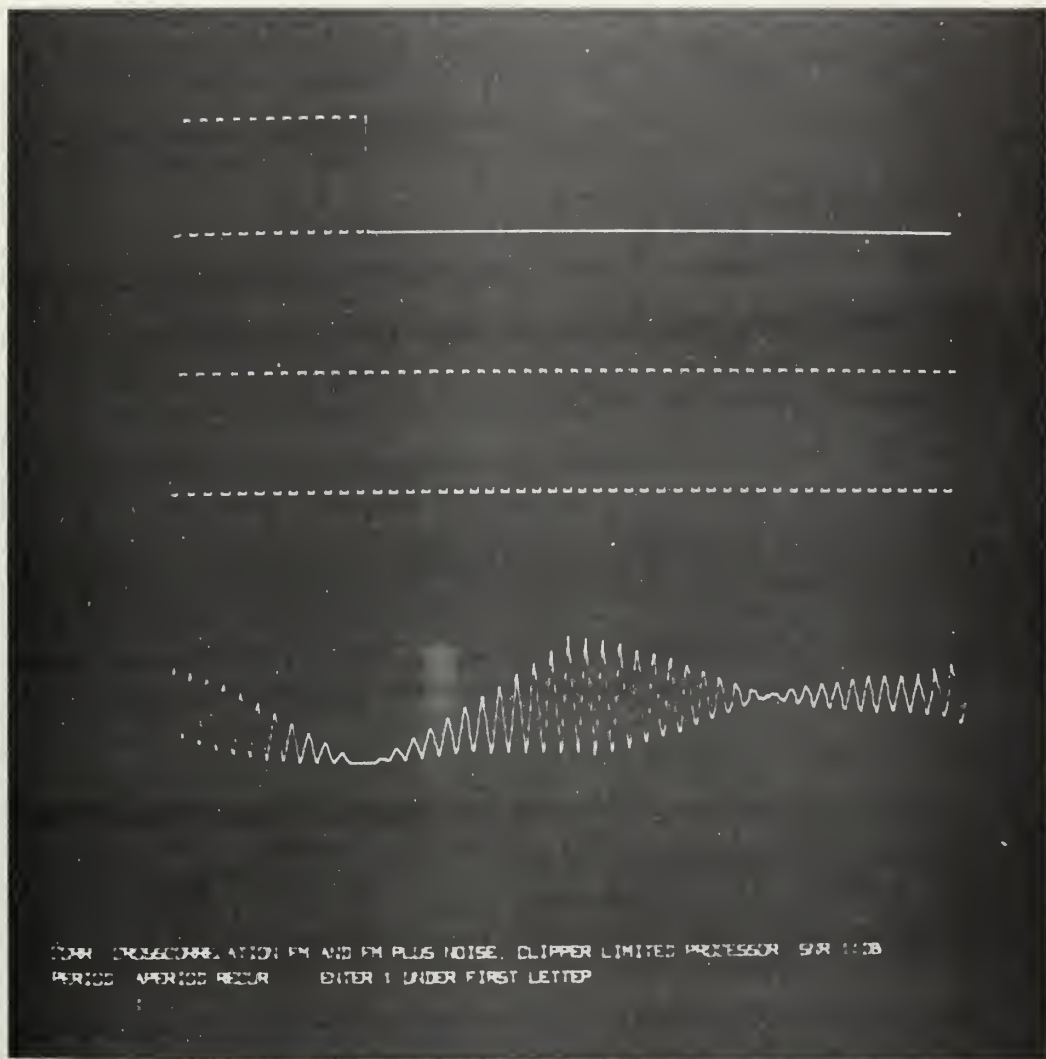


Figure 1.21. This figure shows the result of processing the signals of Fig. 1.19 with a clipper-limited processor.

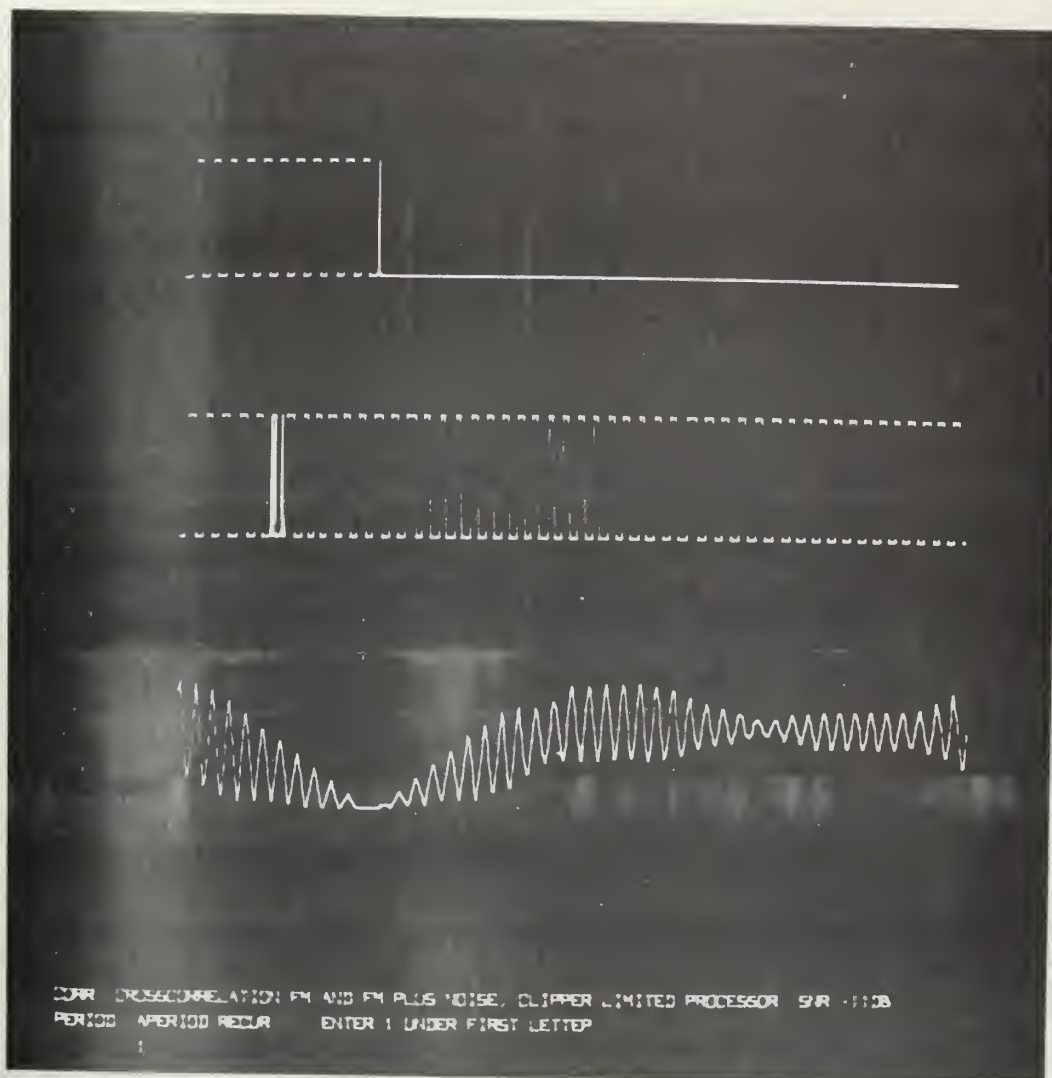


Figure 1.22. This figure presents the results of applying a clipper-limited processor to the signals of Fig. 1.20, where a low signal-to-noise ratio was employed. Since the amplitudes are displayed normalized to the same value, comparison can be made by considering the peak-to-peak oscillations of the outer segments. At low SNR the non-linear processor approaches the linear processor better than when the SNR is high.

For the next experiment (Fig. 1.23) in order to simulate a doppler-shifted reception, the FM signal was modified, but not the replica. The input signal-to-noise ratio is the same as for the case of Fig. 1.19 in order to present the degradation caused by doppler.

In Fig. 1.24 a replica of a propeller signature is cross-correlated against a long sequence of incoming samples where the desired signal is present but at a very low signal-to-noise ratio. The replica was correlated against the long sequence using the partition method; the result displayed is only a section of the correlation corresponding to 9.12 msec. It was observed that the peak output was low, although the oscillatory character of the correlation seemed to carry much of the information required for detection. A Fourier transform of the result, illustrated in Fig. 1.25, showed how this information could be identified.

The last figure of this series, Fig. 1.26 is the result of an experiment similar to that of Fig. 1.20 but with the sampling rate just greater than twice the bandpass (up to this point the sampling rate used was twenty times the highest frequency). It was noted that the displayed replica is completely distorted (no narrow band filters were used to recover the signal from the samples) but revealing an amplitude variation in the order of the bandwidth. The cross-correlation, although of unrecognizable shape, has a highest peak. In this case, where both the total number of samples and the bandwidth have been conserved, the reduction in the sampling rate corresponded to a larger integration time with less redundant samples being processed. Conversely, in the case of Fig. 1.20 the integration time was short and several non-independent samples were processed, contributing nothing to the result.

The importance of input-output performance and ROC determination for comparison of processors and transmission codes was verified by several problems simulated as described in a later section.

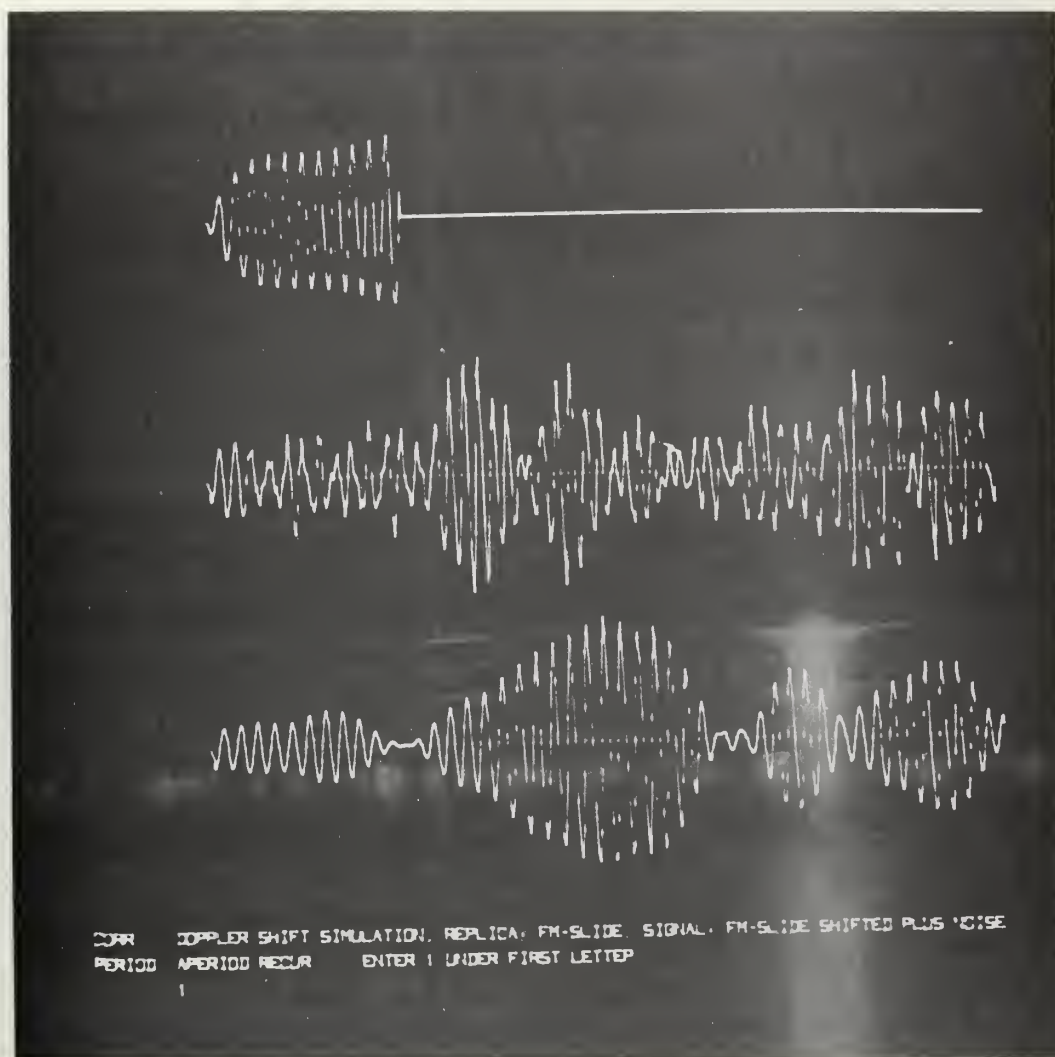


Figure 1.23. This figure presents the result of an experiment designed to simulate a doppler shift of the incoming signal. Comparison with Fig. 1.19 demonstrates the degradation in the correlation as a consequence of the shift.

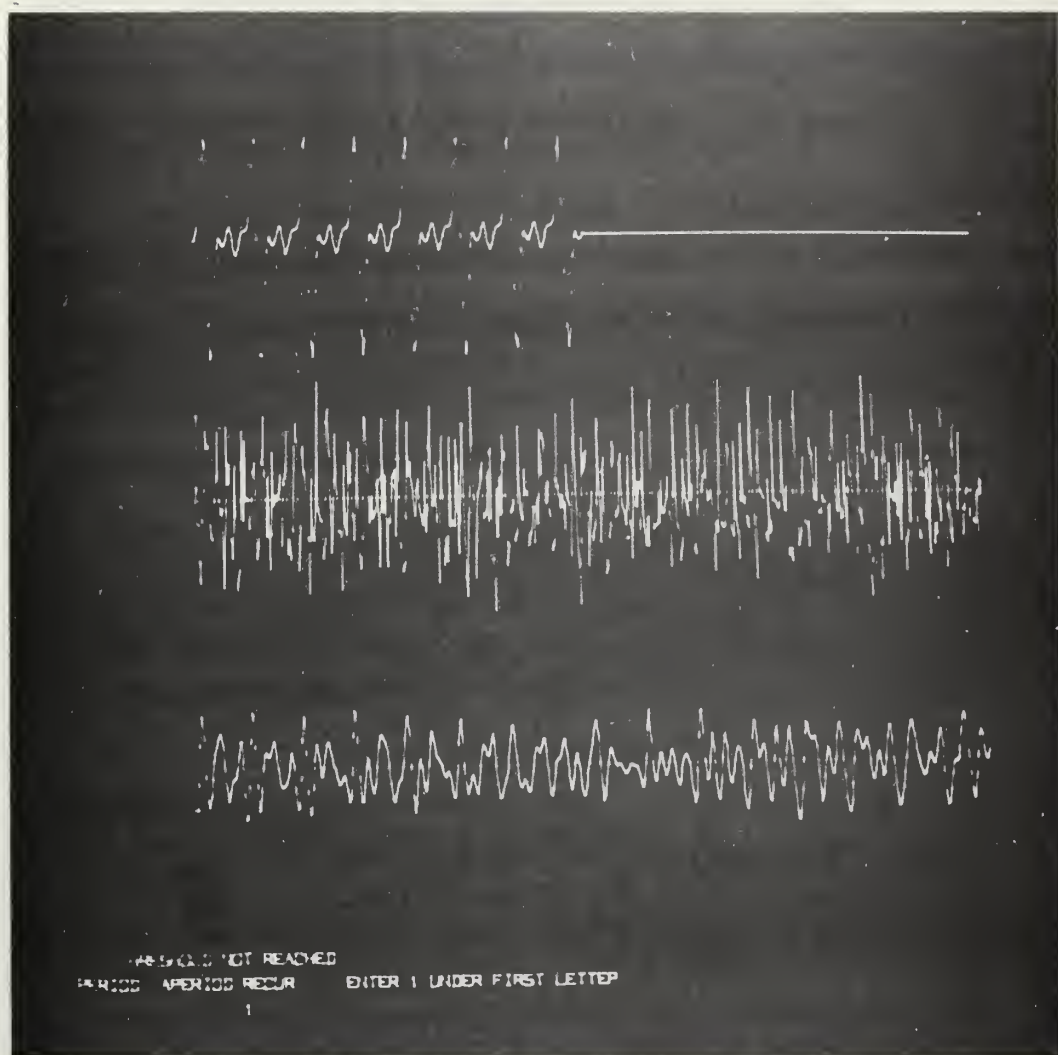


Figure 1.24. In this picture a section of the cross-correlation of the propeller noise replica and the continuously received signal with large amount of noise is shown. Note that the cross-correlation function, if not further processed, does not reveal any information.

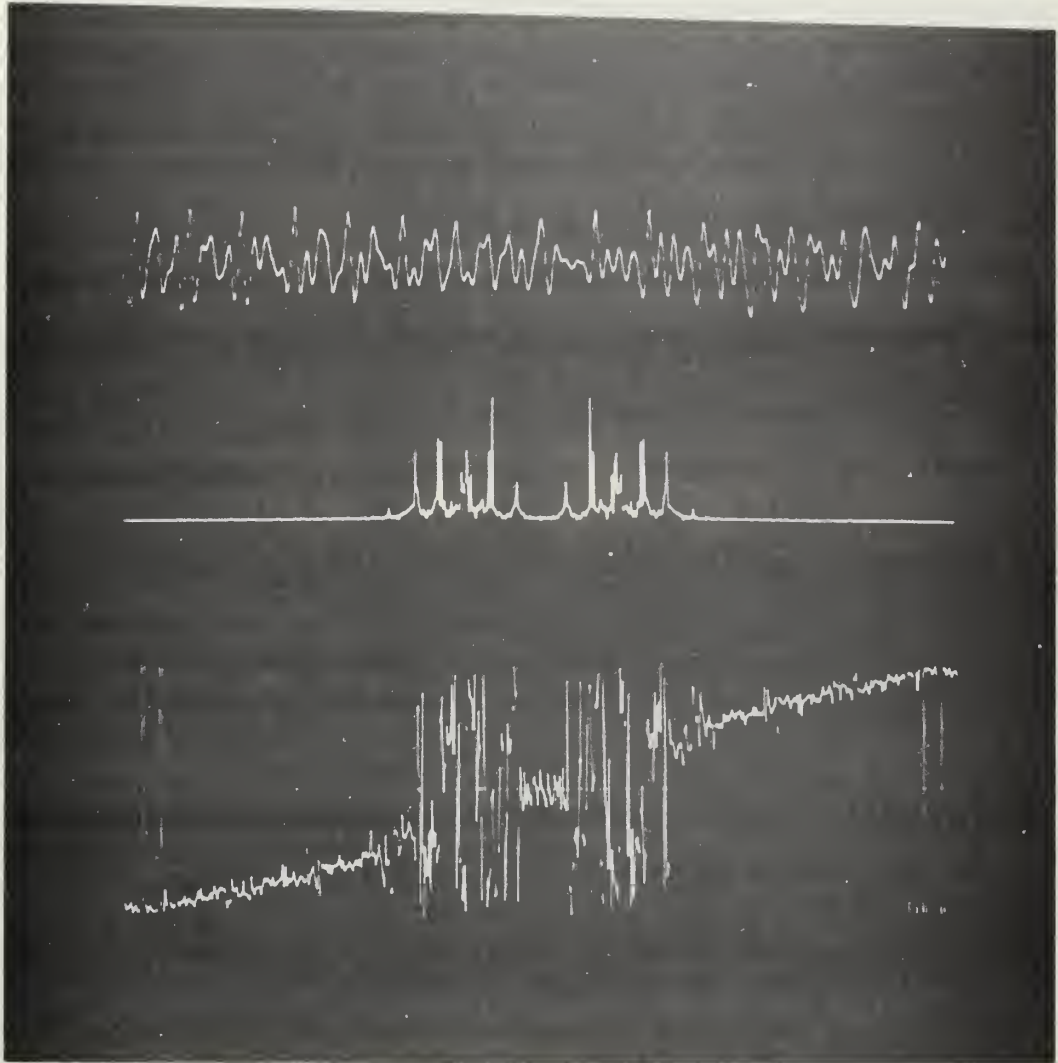


Figure 1.25. This figure shows the result of the spectral analysis of the cross-correlation function displayed in Fig. 1.24. Comparison with Fig. 1.8 demonstrates that the presence of the original signal is now readily identifiable. The same is not true, however, in the power spectrum of the signal before correlation (Fig. 1.13).

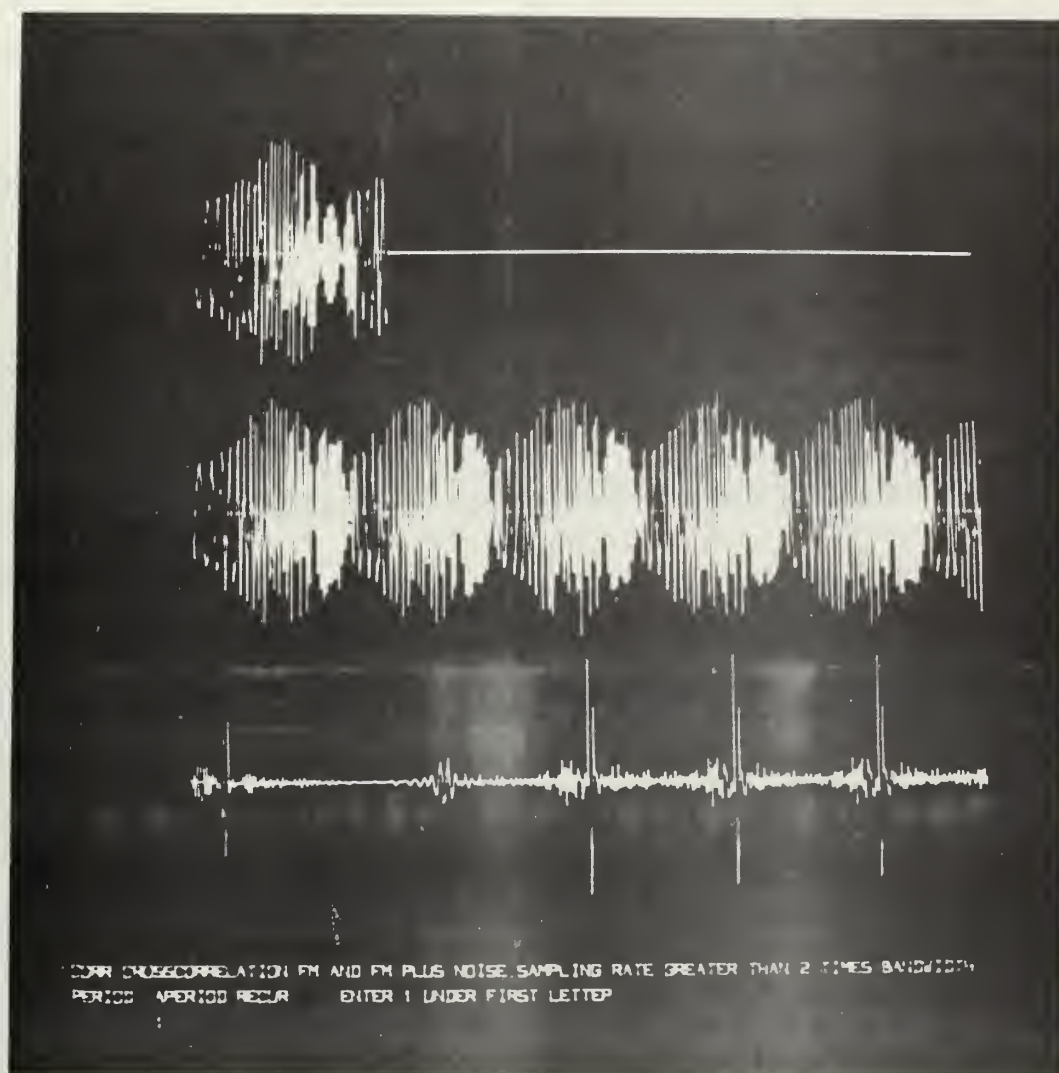


Figure 1.26. This figure presents the effect of sampling the signals of Fig. 1.20 at a sampling rate close to twice the signal bandwidth. Note that while the signal is badly distorted at this low sampling rate, and without any reconstruction filtering, the peak of the cross-correlation is much better defined.

G. DETAILED DESCRIPTION

In the following sections a detailed description of the system implementation together with a review of the mathematical concepts utilized will be given. Further, a brief section on the graphics terminal and discussion of sample simulations is presented. Finally, some conclusions and recommendations for additional applications are provided.

The author does not claim originality for the theory discussed, which can be found in good reference books such as those mentioned in the text. Perhaps only a certain degree of innovation exists in the interpretation given to information bandwidth which seems to depart somehow from the conventional.

The work described here was, and was intended to be, an engineering study. A program requiring a solid theoretical background, a clear view of the objectives, and a willingness to make the unavoidable compromises characteristic of real problems, in order to transform the resources available into a useful product.

II. DIGITAL PROCESSING

The system was implemented to process the signals digitally, since the great majority of processors of interest are digital. The reason for that rests mainly upon the ability of digital machines to handle algorithmic operations very rapidly and with high precision. The discretization of data required for digital processing does not necessarily represent a reduction in information. If the sample rate is properly adjusted, only independent samples may be processed and no time is wasted with the redundancy existing in the original signal.

To use the XDS-9300 of the laboratory it was necessary to write programs in order to simulate the system and processors. Since this is a general purpose computer several other processors can be simulated without any hardware change, requiring only the addition of new programs or modification of those initially included.

In this section a description of the computer program, its usage, and the theoretical reasons determining its construction are presented together, in order to make it easier for the reader to follow the discussion.

A. OVERLAY STRUCTURE

Since, as noted before, the computer has to store in core a large collection of library subroutines to be used in the program, including those to control the analog computer and the graphics terminal, and also has to store relatively large arrays for the computations of correlations and convolutions, little space is left to the program itself. To overcome this difficulty the program was segmented as shown in Fig. 2.1.

OVERLAY STRUCTURE

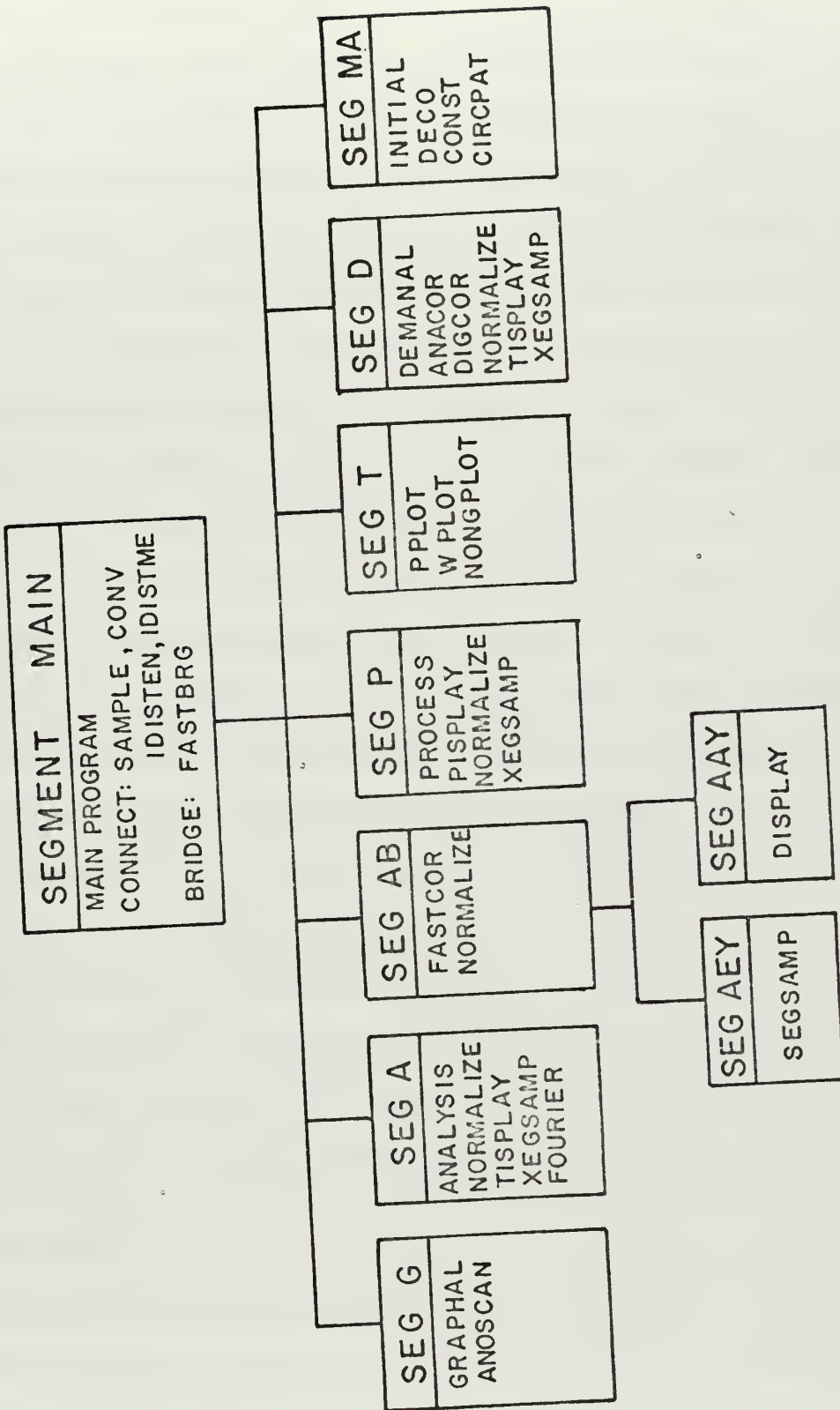


FIGURE 2.1

In brief, the purpose of each segment is as follows. Segment MAIN contains the program initialization and the pivot points where the status is saved when a new overlay is called in. Segment G is composed of the subroutines required to control the analog computer from the graphics terminal. Segment A permits bringing in functions and performing operations on them under operator control, and constitutes the base of the ANALYSIS mode. Segment AB is the principal part of the processor and computes the cross-correlation of two functions; it may be called either from ANALYSIS or PROCESS through a bridge in the main segment. Segment P allows the representation of a processor to be automatically put to work and enters what is called the SIMULATION mode. Segment T is designed to copy what is presented on the screen. Segment D computes correlations through a direct approach for a DEMONSTRATION mode. Finally, segment MA provides graph and test initialization, decodes the instructions typed in by the operator, and selects the path to be followed. This segment provides the capability to select the type of processor or pre-defined signals, and to enter new parameter values.

If the system is to be modified by addition of new segments, the maximum size allowed is approximately 4500 (octal) words per segment. This limit however depends on the addition or deletion of library subroutines and also on the size of the sample as will be noted later.

B. MAIN PROGRAM

The function of the main program, besides serving as a bridge between the various operations and providing initialization of the simulation, is to define permanent location in memory for some constants and the arrays where the samples and results of operations are stored.

The constants are stored in the COMMON area and their meaning will be explained with the subroutines in the Appendix. One exception to this rule is the constant M that defines the sample size; it can be modified by the operator and is initially set to 256, its maximum allowable value. The value of M must be a power of 2 due to the methods used for computation of correlation and Fourier transforms.

The arrays defined in the main program are passed to the subroutines as arguments; with this approach it is readily possible to make use of the same storage area for different applications according to the situation, and it is not necessary to modify the subroutines when a new value is used.

The sample size, M, serves as a parameter to define the blocks of array storage. At this point it is convenient to remember that in the XDS-9300 a real variable uses two words of storage while an integer requires only one. It follows then, that one block of M locations actually requires 2M words.

The map of Fig. 2.2 shows the correspondence of designations throughout the program.

The basic assignment of the arrays is as follows:

YR — replica, or the signal transmitted; size M

YS — signal, or received signal; size 3M, what allows a faster computation of recursive correlation

YC — correlation or result; size 2M

VAGO — empty buffer used for several applications; size 5M

IB — buffer initially designed for text and graph blocks; size 2M
(plus one word)

Before any sample is entered, the main program creates a dummy replica and signal, two different size rectangular pulses which allow easy verification of the system functioning. These two functions are immediately

MEMORY MAP

MAIN	YS	YC	YR	VAGO	I BLOCK
INITIAL CONST					IB
DECO					I L IOP B F
GRAPHAL				IB	
ANOSCAN				I T I I MT X A V A	
ANALYSIS	YS	YC	YR	VAGO	IB
SEGFOUR	YA	YB	YC	FT	IB
DISPLAY	YA	YB	YC	VAGO	I BLOCK
FASTCOR	YA	YB	YC	FF	YD
		FT			FW
					IB
SEGSAMP	YS		YR		
PROCESS	YS	YC	YR	VAGO	IB
PISPLAY	YS	YC	YR	VAGO	IB
P PLOT	XY				IB
DEMANAL	YS	YC	YR	VG	IB
ANACOR DIGCOR	YB		YA	YC	IB
RECOR	YB		YA	YC	YD
					IB

FIGURE 2.2

dumped into secondary storage. This is also always done with the replica, signal and correlation in the ANALYSIS mode. However, only the very last one of each of these functions entered into the system is saved.

C. SUBROUTINES SEGSAMP, XEGSAMP AND SAMPLE

Subroutines SEGSAMP and XEGSAMP have the responsibility of controlling the sampling of the analog computer. They are identical but have distinct identifiers since they appear in two different levels of segmentation. They also provide normalization so that the amplitude of the samples is less than a maximum value compatible with the system.

Another function of SEGSAMP is to prepare the signals for the simulation of three types of processors: linear cross-correlator, amplitude-limited cross-correlator and clipper-limited cross-correlator.

SEGSAMP controls three types of sampling according to the value of INDEX, i.e., replica, initial signal, or continuation signal.

The replica normalization factor, RCOF, signal normalization factor, SCOF, and their ratio Q are saved in common storage, because they are necessary to the computation of the input signal-to-noise ratio, SNR; Q is also necessary to correct the computed threshold value for detection.

The processor type is determined according to ISAMP, i.e., linear, clipper-limited or amplitude-limited.

INDEX has the initial value of one and ISAMP is zero; these numbers however may be set as desired by the operator.

The actual samples are not taken by SEGSAMP. After the subroutine is entered and the proper sampling sequence is selected, the computer is trapped in a loop until an interrupt is generated by the sample clock. At this moment the subroutine SAMPLE is connected and samples the analog

computer, storing the value into ARRAY; at the same time it decreases a counter, MAC, which has the effect of opening the loop in SEGSAMP and allows copying ARRAY into the proper location.

The sample is taken at the circuit node connected to the trunk line determined by LINE whose initial value is one, but can also be adjusted to any legal trunk line designation.

Figure 2.3 is a simplified flow chart that illustrates the operation of the subroutines.

D. SUBROUTINES DISPLAY AND TISPLAY

These are also the same subroutine with different calling names to allow their location in two levels of segmentation.

DISPLAY is designed to encode the functions as required for display on the graphics terminal. It has basically three options. The first option was developed for the presentation of correlation and convolution operations. In this case the y-axis is positioned at the left of the screen. The signals to be correlated and the result of the operation are plotted in three sets of coordinate axes one below the other. As a variation of this option only the first function and the three coordinate axes or only the other two functions are displayed at one time; this feature is useful to the subroutine FASTCOR as is shown later. The number of points used to describe each function may also be varied. The second option is applied to the Fourier transform and here the same number of points are required for all three functions. The y-axis is placed at the center of the screen. In this case the first one is the time function itself, the second the absolute value of the transform, and the third the phase angle. The last option is provided for single function display. The function is presented at the center of the screen with the ordinate expanded.

FLOWCHART



FIGURE 2.3

E. SUBROUTINE FASTCOR

This subroutine constitutes the basic feature of the system and its purpose is to perform a fast convolution. Next will be presented a review of the mathematical concepts that were considered to build this program.

The convolution of two functions is defined as [Ref. 4 and 8]:

$$f(\tau) = r(t) * s(t) = \int_{-\infty}^{\infty} r(x)s(\tau-x)dx = \int_{-\infty}^{\infty} r(\tau-x)s(x)dx$$

The cross-correlation is:

$$R(\tau) = \int_{-\infty}^{\infty} r(x)s(x-\tau)dx = \int_{-\infty}^{\infty} r(x-\tau)s(x)dx$$

This is a similar integral to the convolution with the difference that now none of the functions is time reversed.

If one of the functions is taken, say $r(t)$, and time reversed before calculating its convolution

$$g(\tau) = r(-t) * s(t) = \int_{-\infty}^{\infty} r(x-\tau)s(x)dx$$

then

$$g(\tau) = R(\tau)$$

and it is concluded that the evaluation of both integrals reduces to a single problem, since one of the functions can always be reversed prior to taking the product and integrating.

The solution of these integrals is by no means a simple task. Even in the case where both functions are limited in time so that the integral vanishes outside a certain interval, that is, when it is only necessary to evaluate

$$\int_{x_1}^{x_2} r(x)s(x-\tau)dx$$

the computation is not straight-forward.

In practical applications this integral is normally evaluated by graphical or numerical methods. In both cases the approximate solution is still time consuming in general.

Another method to evaluate the convolution integral can be reached through the theorems of Fourier transformation.

Fourier transform and its inverse are defined as:

$$F(\omega) = F[f(t)] = \int_{-\infty}^{\infty} f(x)e^{-j\omega x}dx$$

$$f(t) = F^{-1}[F(\omega)] = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega$$

The inverse transform of the product of the transforms of each of the functions $r(t)$ and $s(t)$ is as follows:

$$\begin{aligned} F^{-1}[R(\omega)S(\omega)] &= \frac{1}{2\pi} \int_{-\infty}^{\infty} R(\omega)S(\omega)e^{j\omega\tau}d\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} r(x)e^{-j\omega x}dx \right] S(\omega)e^{j\omega\tau}d\omega \\ &= \int_{-\infty}^{\infty} r(x)dx \left[\frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega)e^{j\omega(\tau-x)}d\omega \right] = \int_{-\infty}^{\infty} r(x)s(\tau-x)dx \end{aligned}$$

Since, as is easily seen, the Fourier transform of $r(-t)$ is $R^*(\omega)$, it immediately follows that

$$F^{-1}[R^*(\omega)S(\omega)] = \int_{-\infty}^{\infty} r(x-\tau)s(x)dx$$

What these expressions denote is that if the Fourier transform of both functions is known, the evaluation of the convolution or correlation integrals may be obtained by multiplying the transforms and then performing the inverse transformation. This process greatly simplifies the task if there is a method to easily compute Fourier transforms.

Such methods are not known, in general. However if the functions are represented in the discrete domain, there are fast solutions that can be obtained with digital machines. The computation then gives very close estimates for correlation and convolution.

In the discrete case — the functions are represented as $r'(t)$ and $s'(t)$ thus [Ref. 5]:

$$r'(t) = \sum_{n=-\infty}^{\infty} r(nT) \quad \text{and} \quad s'(t) = \sum_{n=-\infty}^{\infty} s(nT)$$

where T is the spacing between samples, and if the values of $r(t)$ and $s(t)$ are zero outside a certain interval, say 0 to $(N-1)T$ as always happens in practical situations,

$$r'(t) = \sum_{n=0}^{N-1} r(nT) \quad s'(t) = \sum_{n=0}^{N-1} s(nT)$$

The discrete Fourier transform is defined as

$$R(k\Omega) = \sum_{n=0}^{N-1} r(nT)e^{-j\Omega Tnk}$$

with $\Omega = \frac{2\pi}{NT}$ and an inverse transformation

$$r(nT) = \frac{1}{N} \sum_{k=0}^{N-1} R(k\Omega)e^{j\Omega Tnk}$$

The discrete Fourier transform has several properties, many of them paralleling those of the continuous transform. One fundamental distinction, however, results from the definition of Ω : the transform or the inverse transform has only N distinct values for k (or n) between 0 and $N-1$ with these values repeating themselves for k outside this range. This may be seen as the transformation giving the result of a periodic sequence with period N .

A similar relation exists between the convolution and the product of the transform of two functions as occurred in the continuous case:

$$\begin{aligned}
 f(nT) &= \frac{1}{N} \sum_{k=0}^{N-1} R(k\Omega) S(k\Omega) e^{jnkT\Omega} \\
 &= \frac{1}{N} \sum_{k=0}^{N-1} \left[\sum_{\ell=0}^{N-1} r(\ell T) e^{-j\ell kT\Omega} \right] \left[\sum_{m=0}^{N-1} s(mT) e^{-jmkT\Omega} \right] e^{jnkT\Omega} \\
 &= \frac{1}{N} \sum_{\ell=0}^{N-1} \sum_{m=0}^{N-1} r(\ell T) s(mT) \left[\sum_{k=0}^{N-1} e^{j(n-m-\ell)k\Omega} \right]
 \end{aligned}$$

It is not difficult to show that the summation inside the brackets is zero unless m is equal to $(n-\ell)$ modulo N when the result is zero. Hence, substituting $(n-\ell)$ modulo N for m :

$$f(nT) = \sum_{\ell=0}^{N-1} r(\ell T) s((n-\ell)T)$$

where $(n-\ell)$ is to be considered as $(n-\ell)$ modulo N . $f(nT)$ has then an expression similar to that given for $f(\tau)$ and it may be said that $f(nT)$ is a convolution. However it has the special characteristic of repeating itself with period N and so is named a circular or periodic convolution.

As a conclusion, an estimate of the convolution of the functions $r(t)$ and $s(t)$ is $f'(t) = \sum_{n=0}^{N-1} f(nT)$, where r and s are assumed periodic with period NT .

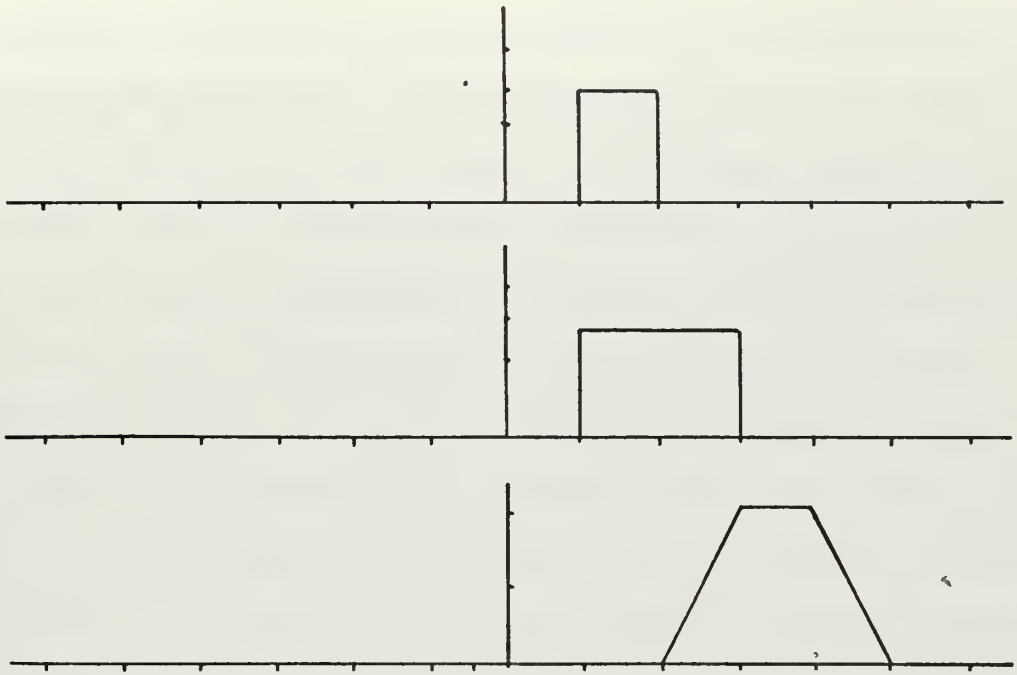
To evaluate that expression it is possible to take the discrete Fourier transform of each function, multiply them together and then compute the inverse transform.

Fortunately, a very convenient and fast algorithm known as Fast-Fourier-Transform has been developed and is specially adapted for use with digital machines [Ref. 3].

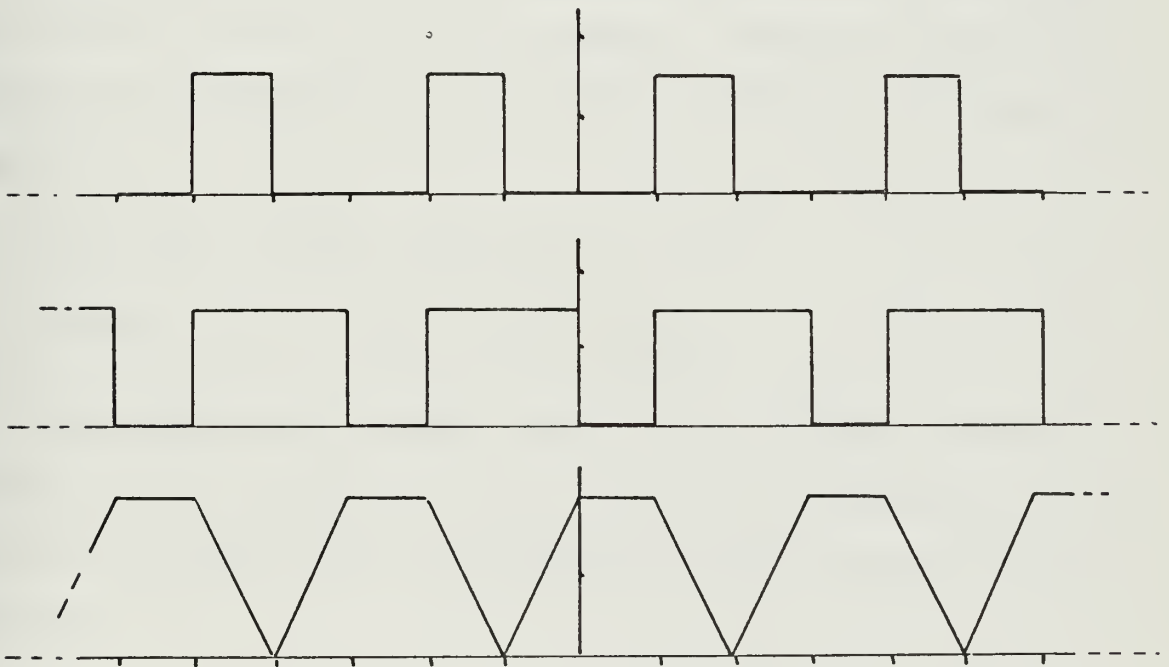
Thus, it is now possible to evaluate convolutions and correlations for practical, real-time applications.

But not all functions dealt with are periodic. On the contrary, very often they are limited to a relatively short interval of time. The method can still be applied if the functions are prepared before performing the convolution. Figure 2.4 illustrates how this can be done. The convolution of two rectangular pulses is shown in Fig. 2.4a and in 2.4b the same thing is presented in the case of periodic repetition of the pulses. The periods were purposely chosen to space the pulse widely so that, when one series is slid across the other, there is no possibility of any single pulse of the first function coinciding with more than one pulse of the second function at the same time. One period of the convolution has then the same form as the convolution of the aperiodic functions. Therefore, in order to evaluate the aperiodic convolution it is necessary only to extend both functions with zeros, constructing periodic functions with a period size of $M+N$, where M and N are the actual sample sizes of the original functions, and taking only one period from the resulting periodic convolution.

CONVOLUTION



a) Aperiodic



b) Periodic

FIGURE 2.4

Another application of this method refers to the case when one of the functions is very short in comparison with the other, as appears in a detection problem when a short replica is correlated against a very long incoming signal. This situation is presented in Fig. 2.5.

The long sequence is partitioned in sections of size M , each section extended with N points from the previous section, and the small one extended with $M-N$ zeros. If the correlation of the N samples (plus M zeros) against successive $N+M$ sized sections is performed, and the results added, it is evident that the first N points of each partial result will be originated in the same way as the last N points of the previous section. This distortion is avoided if the N first points of every partial result are discarded.

In the derivations mentioned above, no restriction was imposed upon the functions that can in general be complex. Therefore in practice a group of four convolutions or correlations is performed at the same time, i.e.,

$$r(t) = r_r(t) + jr_i(t); \quad s(t) = s_r(t) + js_i(t)$$

$$r(t)*s(t) = r_r*s_r - r_i*s_i + jr_r*s_i + jr_i*s_r$$

In most applications, however, real functions are dealt with and it is then possible to perform more than one operation simultaneously. For instance, the partial result can be evaluated for two sections in one operation, or:

$$r(t) = r_r + j0 \quad s_1(t) = s_{1r} + j0 \quad s_2(t) = s_{2r} + j0$$

$$s_s = s_{1r} + js_{2r} \quad r(t)*s_s(t) = r_r*s_{1r} + jr_r*s_{2r}$$

After the operation is concluded the real part is taken as one section of the result and the imaginary part as the other.

PARTITIONED CONVOLUTION

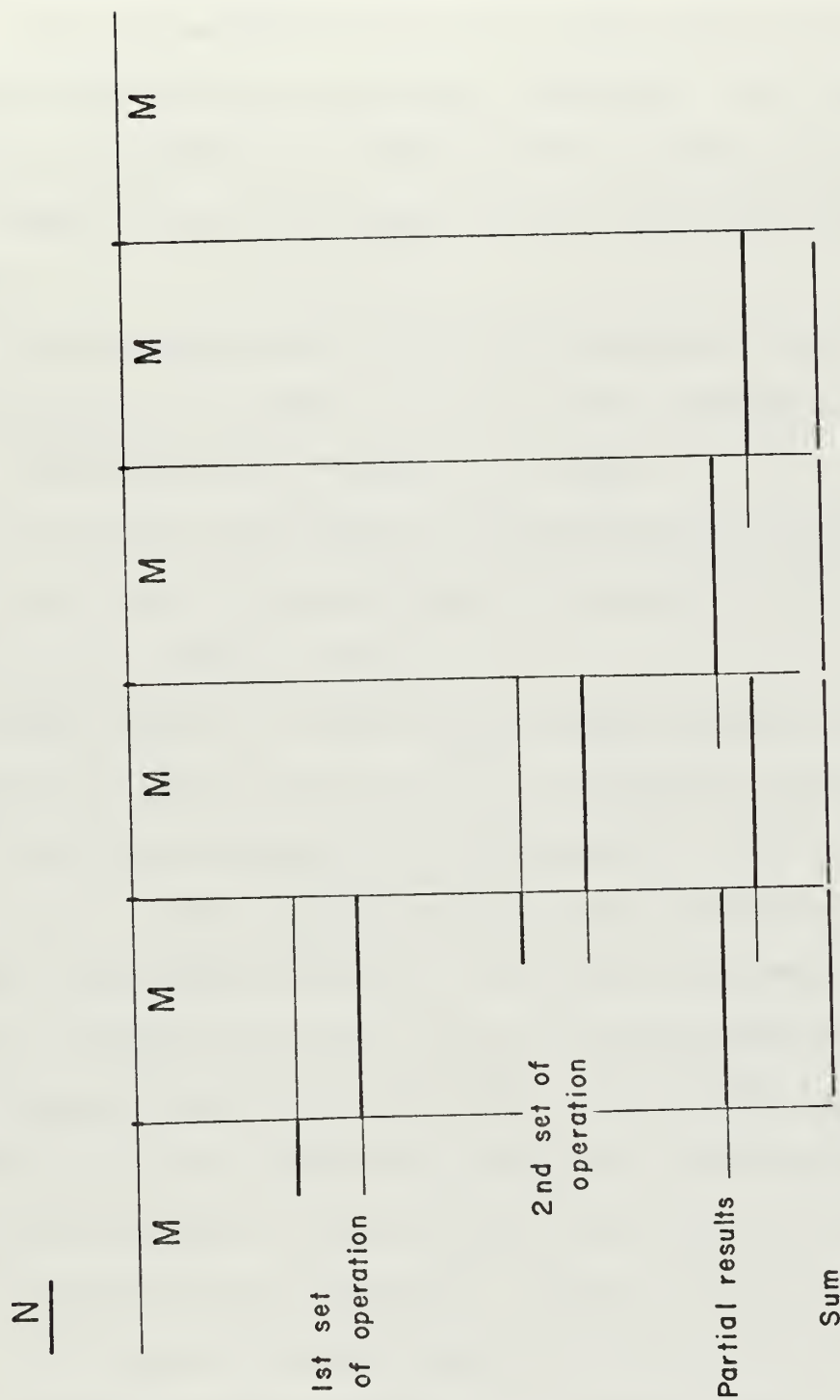


FIGURE 2.5

The subroutine FASTCOR uses for the evaluation of the Fourier transform a very fast program, FOUR2, based on the Cooley-Tuckey method, which is available in the library of the Computer Laboratory. This program requires that the functions to be transformed have a number of points equal to a power of 2, and is the reason for this restriction on the sample size.

FASTCOR has three main options either for convolution or correlation. They are assigned by the argument IT, i.e., periodic, aperiodic, or recursive. Figure 2.6 shows a flowchart for FASTCOR.

Initially the size of the samples is checked against the option selected. In any case, the prepared sample (a "period") for both samples has to have the same number of points.

If a periodic operation is chosen and the second function is longer than the first one, with a maximum of 512, it is reduced to the shorter length. If an aperiodic operation was selected and the sizes of both samples are equal, in which case the maximum length is 256 (M), the two functions are padded with zeros to 2M. In case the first sample is smaller, it is limited to one-half of its size and the second one has the number of points reduced to a value equal to its original value minus one-half of the first function's original size. Both functions are then padded with zeros up to the initial size of the second function; this assures no overlapping in the result. Finally, with the recursive option the first function is always assumed to be smaller than the second which has a maximum acceptable length of 512; the first sample is padded with zeros to the length of the second.

The next step is to flip the first function, send it to be displayed on the screen, since it will soon be erased from memory, and store it as

SUBROUTINE FASTCOR SIMPLIFIED FLOWCHART

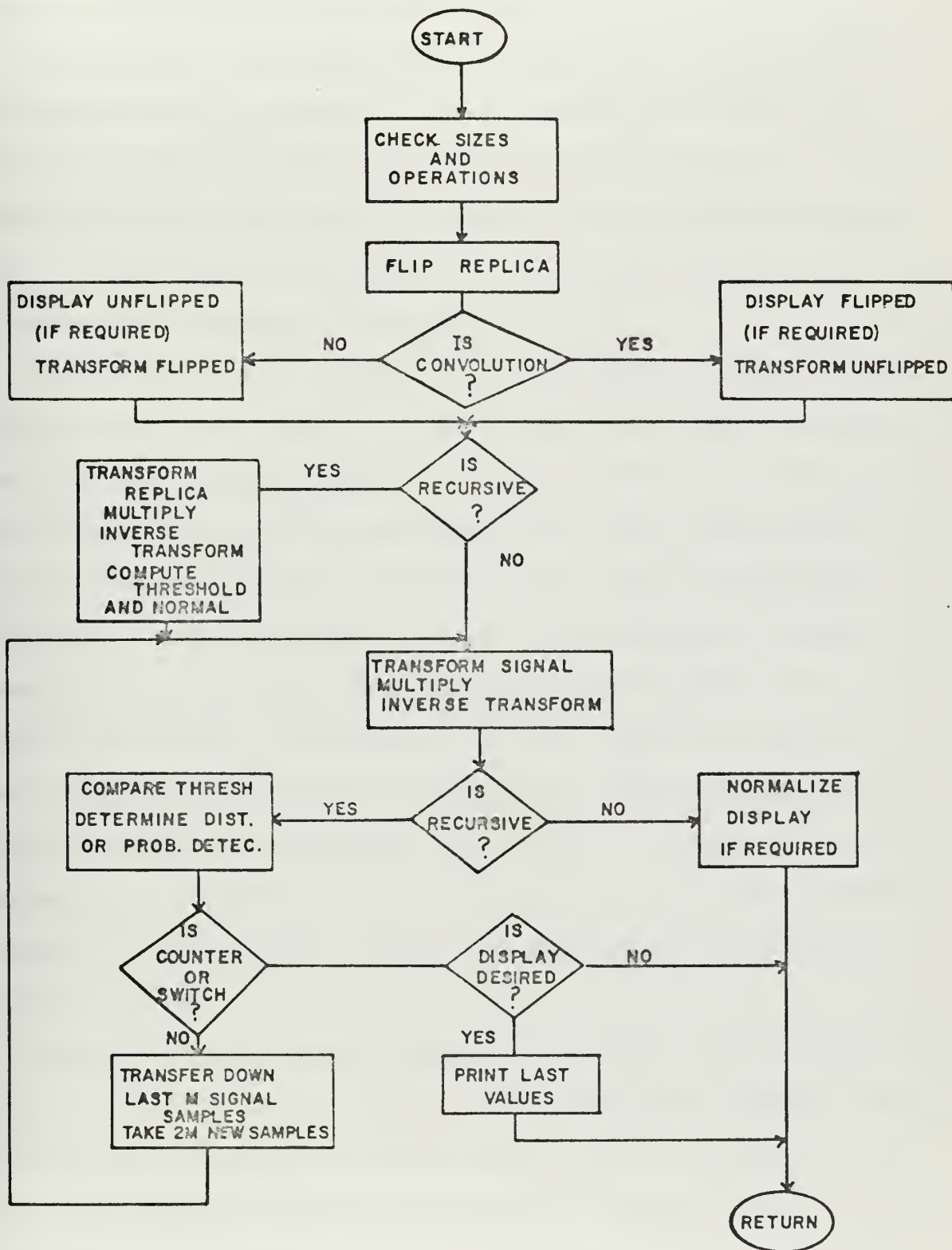


FIGURE 2.6

complex with imaginary part zero. If a correlation is to be performed, the function is displayed without reversal but reversed and stored as complex; for convolution the opposite occurs.

If the option is not recursive, the second function is also stored as complex with zero imaginary part, and both are then Fourier transformed, multiplied together, and the result inverse transformed. During these operations some transfers of storage occur to minimize the space used. Finally, the second function and the result are shown on the screen unless no display is required.

When the option is recursive, an auto-correlation is initially performed on the first function to establish the normalization coefficient and the actual threshold value. This normalization factor is computed, since the maximum amplitude a correlation can attain occurs when the two functions are identical and therefore corresponds to the maximum value of the auto-correlation. But since the amplitudes of the two functions have been previously and independently normalized, the new normalization factor is corrected by the ratio Q mentioned earlier. It was chosen to normalize the threshold value to the maximum amplitude of the auto-correlation, which means that a relative threshold of one would correspond to a perfect matching of the functions. This actual value of threshold is then further normalized, in the same way as the result, for display purposes.

Having completed this task, the second function is then stored as complex. In the strictly recursive case a double-sized (2M) sample is stored as real, and the next section, with overlap, as imaginary. For the determination of probability of detection, however, a sample section padded with zeros is stored as real, and the next section as imaginary

so that effectively a double aperiodic correlation is taken at one time. After correlation, the two sections are moved in core, compared against the threshold, and displayed if so desired.

If a correlation is being evaluated, the threshold level is also displayed, and an indication appears as to whether this amplitude is reached or not. Information also comes to the screen about the probability of detection that has meaning only if this option is in execution. The last measured input signal-to-noise ratio is also shown.

Every time the threshold is met in a distance measuring operation, the actual distance, represented by the sample number, is displayed. The signal, in this case, is added to noise only at one predetermined distance, i.e., only in one group of samples.

After this phase is completed the first 2M samples of the second function (signal) are discarded and the last M are translated to constitute the new first set. SEGSAMP is then called with INDEX equal to 3 to fill out the function with 2M new samples, and the computation starts again.

Exit from the recursion loop results when either the loop has been executed a preset number of times or when the operator presses a switch at the graphics console.

In all cases the control returns automatically to the sequence indicated by LABEL except when the probability of detection (or the false alarm rate) was requested in the ANALYSIS mode. In that case the probability of detection (or the false alarm rate), threshold setting, number of sections processed and input signal-to-noise ratio are all printed on the line printer, and the operator must command continuation by pressing the RETURN key of the terminal keyboard.

F. SUBROUTINES ANALYSIS AND SEGF0UR

These subroutines belong to the ANALYSIS mode.

ANALYSIS is intended to clarify further the request of the operator and route the computer to the appropriate sequence. The argument IENT is a pointer to indicate whether the subroutine is entered to initiate a sequence or to end it. This is a requirement of the overlay structure since the space it occupies in core has to be released whenever FASTCOR is called.

The initial selection is indicated by IJ, kept in common storage.

IJ equal to one indicates that sampling is desired. Before calling SEGSAMP a reminder is displayed asking the operator if the constants are adjusted to sample the correct function, for the correct processor and from the correct point in the analog (INDEX, ISAMP, LINE). If the answer is negative the constants are displayed to allow corrections after which sample has to be requested again. An affirmative answer initiates the sampling operation and then the new sample is saved in background storage.

IJ equal to two indicates Fast-Fourier-Transform and the operator is requested first to indicate which function: replica or M sized function, signal or 2M sized function, correlation or the result of the last operation performed by FASTCOR. Then SEGF0UR is called to prepare the function selected, perform the transformation through FOUR2, calculate the absolute value and the phase angle of the transform, send them and the original function to the display and read back from secondary storage the functions destroyed during the operation.

When correlation is selected, IJ=3, it is only necessary to indicate if the mode desired is periodic, aperiodic or recursive. FASTCOR is now

called through FASTBRG, a bridge subroutine required to save addresses and parameters while the segment swapping takes place. Back from FASTCOR, FASTBRG re-enters ANALYSIS to restore the functions destroyed and put the result of the correlation in backstorage.

For IJ equal to four (convolution), five (power spectrum), or seven (auto-correlation), the operator must select the function and mode. With convolution, the "other function" is always assumed to be the replica; with the other two operations, recursive is an illegal choice and if made results in an error message. The functions are then moved in storage so as to send to FASTCOR the appropriate parameters. Returning to ANALYSIS, action similar to that for correlation is taken, except when power spectrum is the operation.

The evaluation of power spectrum is done through the application of the Wiener-Kinchin relations [Ref. 8].

It was earlier recalled that the convolution is the inverse Fourier transform of the product of the transform of the functions to be convolved and that the correlation can be treated as the convolution taken with one of the functions time-reversed prior to the operation. Also the transform of $r(-t)$ is equal to the complex conjugate of $R(\omega)$. Now, the auto-correlation of a signal for which the energy content is not finite is defined as

$$P(\tau) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} r(t)r(t-\tau)dt \quad .$$

Hence,

$$P(\tau) = \lim_{T \rightarrow \infty} \frac{1}{T} [r(t)*r(-t)]$$

$$\begin{aligned}
P(\tau) &= \lim_{T \rightarrow \infty} \frac{1}{2\pi T} \int_{-\infty}^{\infty} R(\omega) R^*(\omega) e^{j\omega\tau} d\omega \\
&= \lim_{T \rightarrow \infty} \int_{-\infty}^{\infty} |R(2\pi f)|^2 e^{j2\pi f\tau} df \\
&= \int_{-\infty}^{\infty} G(f) e^{j2\pi f\tau} df
\end{aligned}$$

where $G(f)$ is the power spectral density. If the energy content is finite, the energy spectrum is considered instead of the power spectral density. Therefore, since the power (or energy) spectrum and the auto-correlation form a transform pair, the spectrum may be evaluated by computing the auto-correlation and taking its Fourier transform.

ANALYSIS then determines the auto-correlation to be performed and when re-entered, calls SEGFOUR to transform the result. (None of these results are saved.)

The last control of the subroutine corresponds to IJ equal to six or the display selection. After the answer for the function is selected and decoded the subroutine DISPLAY is called to show that function.

ANALYSIS has to keep track of the number of samples related to each selection and controls the allowed limit for all situations reducing the size of the sample if necessary. The common location II is used for this purpose during the overlays substitution. The size of the last result (CORR) stored is saved in IMEM. Its value is set when the subroutine is re-entered and corresponds to the number of points displayed as indicated by LBG(3).

The flowchart for this subroutine is shown in Fig. 2.7.

SUBROUTINE ANALYSIS
SIMPLIFIED FLOWCHART

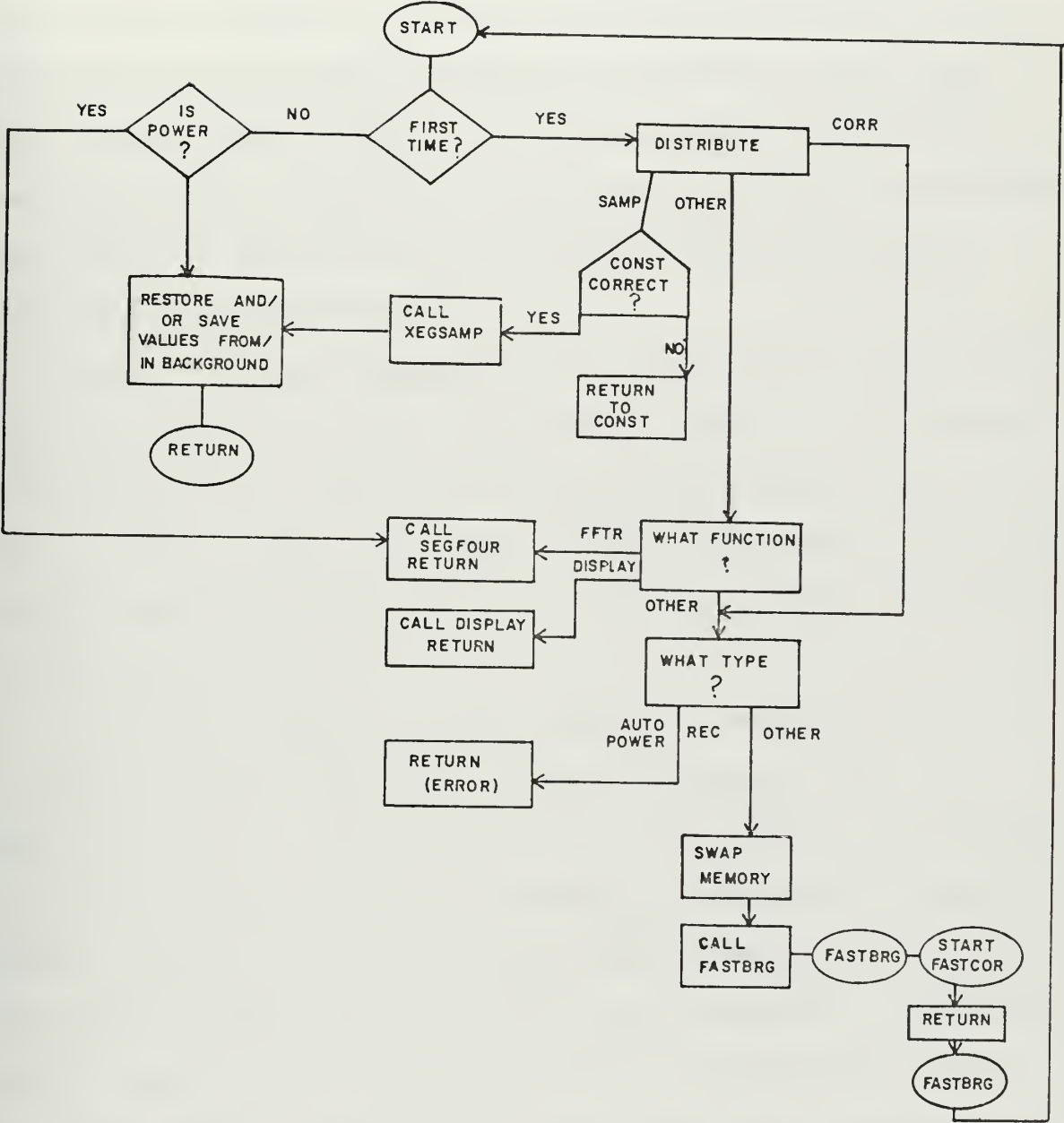


FIGURE 2.7

G. SUBROUTINES PROCESS AND PISPLAY

PROCESS is constructed to chain various sequences in order to execute the operations necessary to compute input signal-to-noise ratio, evaluate the processor input-output performance, and determine the ROC curves. It also permits simulating a situation of target detection and distance measurement. This automatic procedure constitutes a part of the SIMULATION mode except for the signal-to-noise calculation that may be called separately in the ANALYSIS mode.

Except for distance measurement, the program establishes loops where some parameters are varied for every recursion until a limit is reached. Programwise, however, these loops are interrupted several times by transfer to another overlay. More common storage is then required to save the status during the segment swapping, and to allow resumption of the loop in the proper position.

A simplified flowchart for these sequences is presented in Fig. 2.8.

When signal-to-noise ratio is called in the ANALYSIS mode, it is initially necessary to scan the signal and noise injection potentiometers, save their settings, and restore them after the computation is completed. Entered the first time for performance or ROC determination, it determines the number of steps that will be used to calculate the SNR. The operation really starts by saving the processor status and establishing a linear processor, since the input signal-to-noise ratio (after the transducer) is desired; the processor is reset after the computation. Two loop paths are then executed, one to determine the signal power at that level of injection (no noise present), and the other for noise power (no signal present). The ratio, in decibels, is taken and stored as SNR, and displayed if the operation CONST is selected next.

SUBROUTINE PROCESS
SIMPLIFIED FLOWCHART

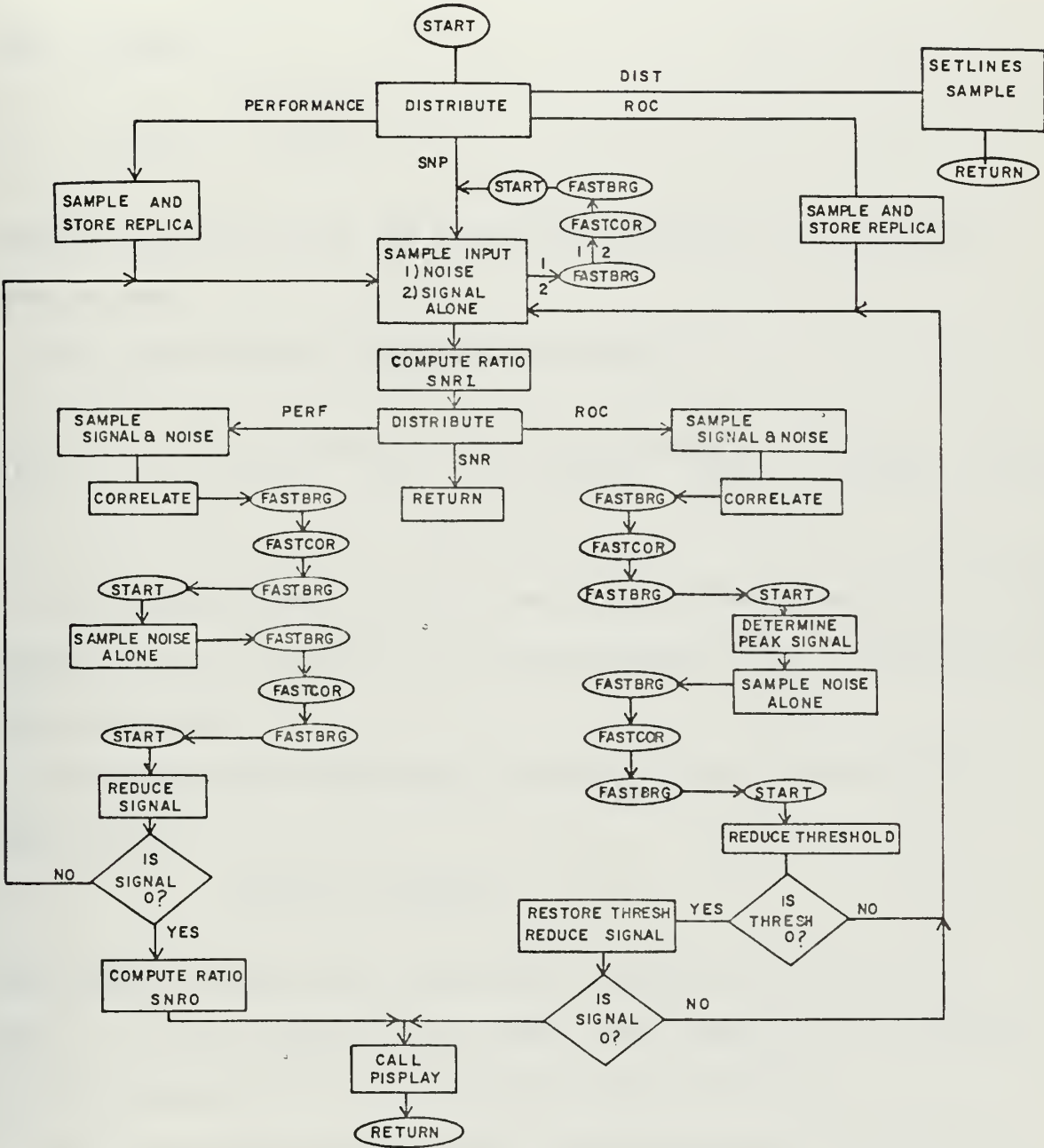


FIGURE 2.8

The Wiener-Kinchin relation again:

$$P(\tau) = \int_{-\infty}^{\infty} G(f) e^{j2\pi f\tau} df$$

When τ is zero

$$P(0) = \int_{-\infty}^{\infty} G(f) df$$

meaning that the value of the autocorrelation when $\tau = 0$ is the total average power.

The absolute value of the autocorrelation is

$$|P(\tau)| = \left| \int_{-\infty}^{\infty} G(f) e^{j2\pi f\tau} df \right| \leq \int_{-\infty}^{\infty} G(f) e^{je\pi f\tau} df = \int_{-\infty}^{\infty} G(f) df = P(0).$$

So, the power content can be found by taking the maximum value of the autocorrelation since $P(0)$ is an absolute maximum. This is exactly what the program does.

The input-output performance or processor gain can be defined by the equation

$$SNR_o = f(p,W) SNR_i$$

where SNR_o is peak output signal power to output noise power ratio SNR_i is the input signal-to-noise power ratio, and $f(p,W)$ is a function of the processor and its associated information bandwidth.

The performance operation begins sampling and storing the replica in a position never to be destroyed during the process in this particular case. Then it uses the previously established values for signal and noise injection that correspond to the highest input signal-to-noise

ratio to be considered, and sequentially computes input signal-to-noise ratio, samples noise alone, cross-correlates replica and noise, auto-correlates this output and takes the maximum (output noise power). It then samples signal plus noise, cross-correlates replica and signal plus noise, finds the maximum (signal peak output). Next it reduces the signal injection and correspondingly increases the noise by a pre-determined step and recycles as long as the signal-to-noise ratio is greater than zero. During the operation the individual values of signal-to-noise ratio, output noise power, and signal peak-output are placed in common storage. When the loop ends, the computed values are rearranged in core and SNR_0 is computed. Finally, the subroutine PISPLAY is called to display the resulting curve with the abscissa as $SNR_i(db)$ and the ordinate as $SNR_0(db)$. (The input signal-to-noise ratio is measured after the transducer filter.)

One detail must be examined in the output noise power computation. Since the cross-correlation of replica and noise has $2M$ points, it may exceed the maximum limit allowable for autocorrelation if M is 256. The solution adopted was to calculate the autocorrelation in two steps, one-half at a time and then add the two maxima; this is legitimate since the correlation is a linear operation.

For ROC determination, the procedure, up to the input signal-to-noise ratio evaluation and the recursion for the several levels of signal, is identical to the sequence just described, with the exception that the replica is placed in secondary storage. Inside the loop however two other loops exist: the outer one for the threshold value that steps down in 0.1 steps from the initial setting until zero is reached; the innermost loop has two paths, one for the calculation of the probability of

detection (signal plus noise) and the other for false alarm rate (noise only). PISPLAY is also used to present the results on the screen. The abscissa is presented as false alarm probability and the ordinate is false alarm rate (per sample). A listing of the signal-to-noise ratio for the curves presented and normalized threshold are also displayed.

According to this description, instead of the receiver operating characteristic the result is a modified ROC where false alarm rate was substituted for false alarm probability. This modification has more practical significance and is also easy to evaluate in a short period of time while the probability of false alarm is not. These two parameters are related by the noise output bandwidth.

At this point a warning should be made. The above estimations serve as an indication of the probable behavior of a processor and that is the purpose of the system. More precise ensemble average estimations would require lengthy repetitions, and it was not considered worthwhile to incorporate them. However, as will be pointed out later, the basic idea can be used for a much less flexible fast analytical system or for special purpose designs.

Detection and distance measurement form the last option of PROCESS which sets parameters and lines to initiate the procedure, and samples the replica and the initial sections of signal. A line is set to disconnect the pattern generator. ICOUNT is set to allow a maximum distance scan corresponding to 20M samples and ISTF to connect the signal at a distance of 12M sampling intervals.

H. SUBROUTINE CIRCPAT,

The SIMULATION mode is completed with this subroutine which is able to respond to the typed commands and set circuits and parameters to

simulate different transmission signals, such as CW burst, FM slide, FSK and propeller signature.

These simulations are better discussed in the section dealing with applications.

I. AUXILIARY SUBROUTINES

A number of auxiliary subroutines were developed to enhance the utility and convenience of this system for the investigation.

GRAPHAL has already been referred to and permits the control of the analog computer from the graphics terminal. It is entered whenever the command for one of its functions is typed; if this function is to scan the analog, ANOSCAN is called. These subroutines may be modified for general use.

PLOT copies the functions displayed on the screen and has three options to output them. The first is through the library routine VLOT, with identifier changed to WLOT since it was necessary to have it loaded only in the overlay in order to reduce core storage requirements. It plots the curves on the line printer in a one-page sized plot where the x-axis is transverse to the paper. NONGLOT, the new identifier given to the library subroutine LONGLOT for the same reason as above, also prints on the line printer but with the x-axis longitudinal to the paper and a plot occupies a variable number of pages depending on the number of points plotted and on a parameter set by the operator. Through another sequence, PLOT uses the eight channel recorder as the output medium. When selected, it first asks that the recorder be made ready, and when the operator presses the RETURN key it waits 20 seconds before starting the

plot. The information about the potentiometer P011 displayed there indicates the normalized threshold level that, if desired, may be introduced in that potentiometer. For this case, an analog probability counter will deliver pulses to the recorder whenever the correlation curve crosses the threshold value. The value for adjustment of P011, however, is meaningful only if the curve displayed corresponds to a recursive type of correlation.

Subroutine DEMANAL routes the command to other subroutines which perform correlations through a direct approach, that is, by simple interpretation of the integral definition. Those subroutines are: ANACOR which performs a periodic autocorrelation using an analog multiplier and integrator; DIGCOR which evaluates auto or cross-correlation entirely digitally; and RECOR which computes recursive cross-correlation.

Subroutine NORMALIZE is intended to normalize a function to some preset value. The storage address, the maximum value desired and the number of points are the arguments passed to it.

When the program is first loaded or after GRAPHAL is called, INITIAL initializes the text and graph directory and the graphics terminal. It is entered to erase the text blocks after an operation is concluded or to erase the graph blocks after a new selection is made (except if the selection is CONSTANTS or any of the plot options).

J. SUBROUTINES DECO AND CONST

DECO works as the control section of the system, allowing the user to type in his selection which is then decoded and interpreted to channel the actions in the desired direction. It forms a table of control words, composed of the first four letters of the selections available, and when

a command is decoded checks it against that table and sets two pointers, II and IJ, and returns to MAIN. II is then used to select the segment while IJ indicates the correct choice. If no entry exists in the table matching the selection, an error message is displayed and a new order may be entered. DECO is called automatically after the completion of an operation or when an illegal procedure is requested in another subroutine.

The control function is completed with CONST which displays various parameters and pointers and permits the operator to alter them inside allowable limits. The names under which they appear are not necessarily those used in the program, but others more easily remembered. A full discussion of this will be given with the graphics terminal usage explanation.

III. ANALOG CIRCUITS

The analog computer was selected to simulate the signals because they are generated in the same form as they occur in reality. Thus the analysis of existing signals can be made by the simple substitution of the actual signals. In this case the analog computer itself acts as the interface to the system.

Another important feature considered was the relatively large parallel logic capability of the COMCOR Ci-5000 analog computer, which allowed construction of control circuits to enlarge the range of utilization and analog processing.

The analog computer circuits included: one sine generator used as PATTERN GENERATOR, one HARMONIC GENERATOR fed by three independent sine generators, one WHITE-NOISE generator, a SIGNAL COMBINER, a BANDPASS FILTER, an INTERRUPT GENERATOR and some auxiliary circuits.

As a general rule the circuits and their parameters are remotely controlled from the graphics terminal with exception of those few which need patch relocation. All lines that can be set are tied to another test line in order to permit status check from the display.

A. PATTERN GENERATOR

1. Theory

The differential equation $\ddot{y} = \frac{d^2 y}{dt^2} = -\omega^2 y$ represents a sine wave and its solution can easily be implemented with integrators and adders. The mechanization of this equation is presented in the following expressions where the bar means that the variable is a scaled (computer) variable, the α_i 's are amplitude scaling coefficients, β is a time scaling coefficient, and t_c is computer scaled time interval:

BASIC SINE GENERATOR

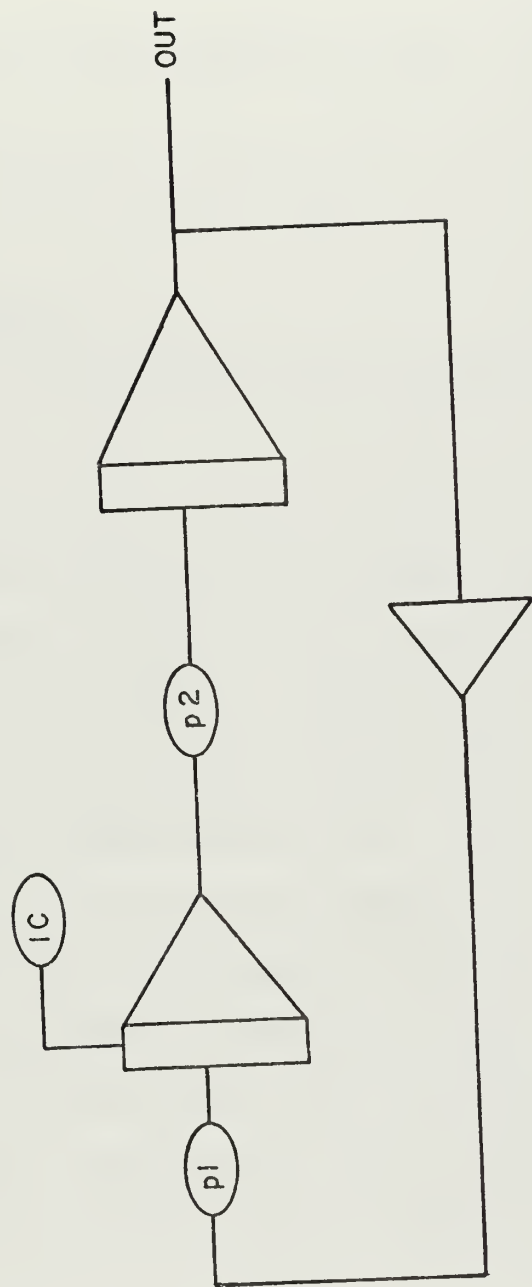


FIGURE 3.1

$$\dot{y} = - \int - \ddot{y} dt - A$$

$$y(0) = -A$$

$$\ddot{y}(0) = \dot{y}(0) = 0$$

$$-\ddot{y} = - \int \ddot{y} dt = - \dot{y} - \omega^2 y$$

$$\dot{y} = \alpha \bar{y}$$

$$\dot{\bar{y}} = \frac{\dot{\alpha} \bar{y}}{\alpha}$$

$$\bar{y} = - \int \frac{\dot{\alpha}}{\alpha \beta} (-\dot{\bar{y}}) dt_c - \frac{\dot{\alpha} \bar{A}}{\alpha \beta} = -p_1 [\int (-\dot{\bar{y}}) dt_c + \bar{A}] \quad t = \frac{t_c}{\beta}$$

$$-\dot{\bar{y}} = - \int \frac{\omega^2 \alpha}{\alpha \beta} (-\bar{y}) dt_c = -p_2 \int (-\bar{y}) dt_c$$

2. Implementation

The basic implementation of the equations above is shown in Fig. 3.1.

Since it was desired to make this circuit simulate a CW burst, as well as a linear FM slide and an FSK codes, a multiplier was utilized for p_1 and p_2 because it allows changing frequency during operation without resetting the amplifiers. Additional circuitry was required to generate the multiplier reference voltage, to select the type of operation, and to switch the generator on or off. The final diagram is presented in Fig. 3.2. A017 is the integrator for y and A021 for $-\dot{y}$. A203 and A204 provide polarity reversal as required. The multiplier M401 implements p_1 and M402 introduces p_2 . Relay R26 connects the reference input of the multiplier either to the ramp generator A025 or to the FSK relay R10. P030 and P031 adjust the lower and upper shift reference using A205 and A206 as loads. R006 is the relay that switches the generator output on or off.

The ramp generator computes the value of $R = I + St$ where R is the reference voltage, I the initial setting and S the slope. P026 establishes I , P025 fixes S and P027 fixes the threshold value for C07

79

which controls the reset time for the ramp. P032 and C04 are responsible for the shift instant. P021 prevents the necessity to set P025 to very small settings and A027 and A030 serve as load to A025 when R26 is open.

To make it possible to run this generator at very high actual frequencies (150 Hz), positive feedback is introduced by P017. The setting of this potentiometer is not critical since the limiter L00 prevents the amplitude from growing.

3. Logic

The logic circuitry required to control the pattern generator, presented in Fig. 3.3, is the result of the logic equations developed in the Appendix.

Lines T007 and T010 are set to control the integrator mode through several drivers.

The NAND gates, comparator C07, delay-flop DF02 and line T011 direct the ramp generator operation. It then follows the general mode except if CW is required when it remains always in reset. During computation the ramp generator is made to reset every time a sample period ends.

C04 actuates the FSK switch R10. Lines T016 and T017 bring the voltage to R26 and R06, respectively.

4. Scaling

The presentation following has the purpose of establishing simple reference tables for quick problem scaling. The final objective is to find the value of R, the reference voltage, and the amplifier gain suitable for each case.

The amplifier gain (G) is a function of the input conductance (I-mmhos) and the feedback capacitor (C-mfd), coefficients p_1 and p_2

PATTERN GENERATOR LOGIC

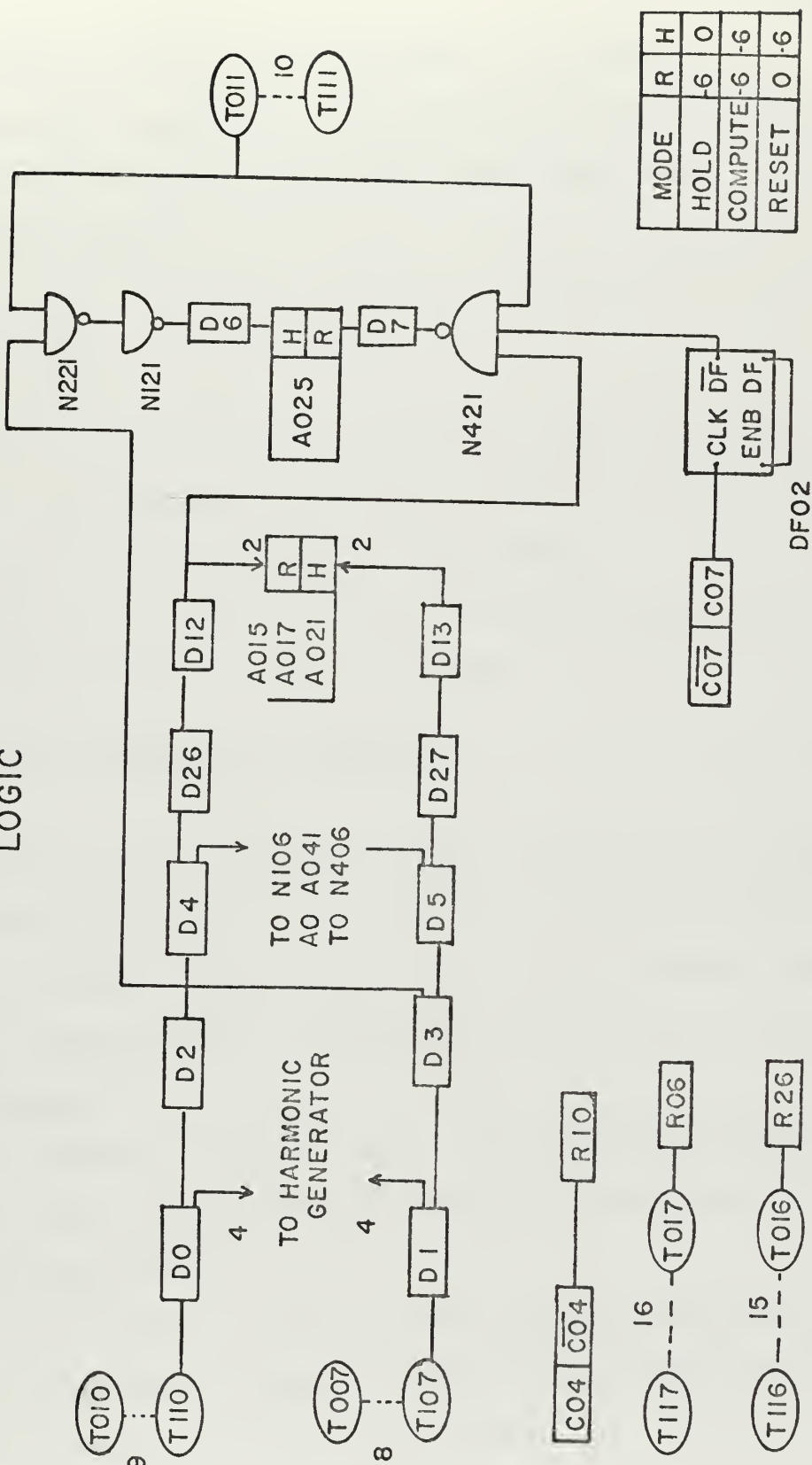


FIGURE 3.3

mentioned earlier are the product RG divided by the constant 100, characteristic of the multiplier. For convenience, R was multiplied by 100 in the final result to become the integer value that is sent to set the potentiometers.

$$P_1 = \frac{RI}{100C} = \frac{\dot{\alpha}}{\alpha\beta} ; \quad \beta = \frac{t_c}{t} = \frac{f}{f_c} = \frac{\omega}{\omega_c}$$

where α and $\dot{\alpha}$ are amplitude scaling factors for y and \dot{y} , β the time scale factor, t_c the computer scaled time interval and f_c and ω_c the scaled frequency and angular frequency respectively.

Since $y = A \sin \omega t$, $\dot{y} = -A\omega \cos \omega t$ assuming A equal 100 for a maximum voltage scaling $\alpha = 1$ and $\dot{\alpha} = \omega$. Thus

$$R = \frac{100 \times 100 C \omega}{\beta I} = \frac{20000\pi f C}{\beta I}$$

R has to be kept below 10000 (100v) and it was chosen to keep C as .1 for real time and .01 for time scale for an actual computer frequency, f_c , around 15 Hz and 150 Hz respectively. The other values follow as a consequence and are tabulated in Table I in the Appendix.

5. Operation

For CW burst code, A027 is kept in RESET through T011, R26 is held in the normally closed position (NC); P025 is set to zero and P026 to the reference voltage R.

In FM slide A027 is made to compute once every sample and then reset. The relay R26 is maintained in the normally closed position (NC). The potentiometers are set in the following way:

P021 to the inverse of the gain of A025;

P025 to the ratio of the upper reference voltage (R_U) minus the lower reference (R_L) to the actual sampling time, i.e.,
 $(R_U - R_L) \times \text{sampling period} \times \text{number of points in the sample}.$

P026 to the lower reference voltage; and
P027 to the upper reference voltage.

For FSK the same adjustments as for FM slide are necessary with the following alterations:

- R26 - placed in normally open (NO) through the action of T016;
- P030 - to the lower reference voltage;
- P031 - to the upper reference voltage;
- P032 - to the central value of the reference voltage R.

In all cases above, if distance measurement is required the relay R06 is maintained in normally open (NO) all the time except at the moment when the burst is to occur when it closes under control of T017.

B. HARMONIC GENERATOR

1. Sine generators

The three sinusoidal oscillators included in the harmonic generator are of the basic type presented in Fig. 3.1, with p_1 and p_2 adjusted by potentiometers. The scaling formulas for the multiplier case also apply here.

The harmonic generator was built to simulate a relatively low-pass line spectra similar to the signal generated by a propeller. One of the sine generators produces the signature while the others produce a "non-white" noise, as if they were other identical sources of slightly different frequency.

2. Multipliers

In the diagram of Fig. 3.4 it is seen that the three generators were combined and fed to four multipliers arranged to give as output, after combination in A052:

$$E = K_1 e + K_2 e^2 + K_3 e^3 + K_4 e^4 + K_5 e^5 + K_6 e^6$$

HARMONIC GENERATOR

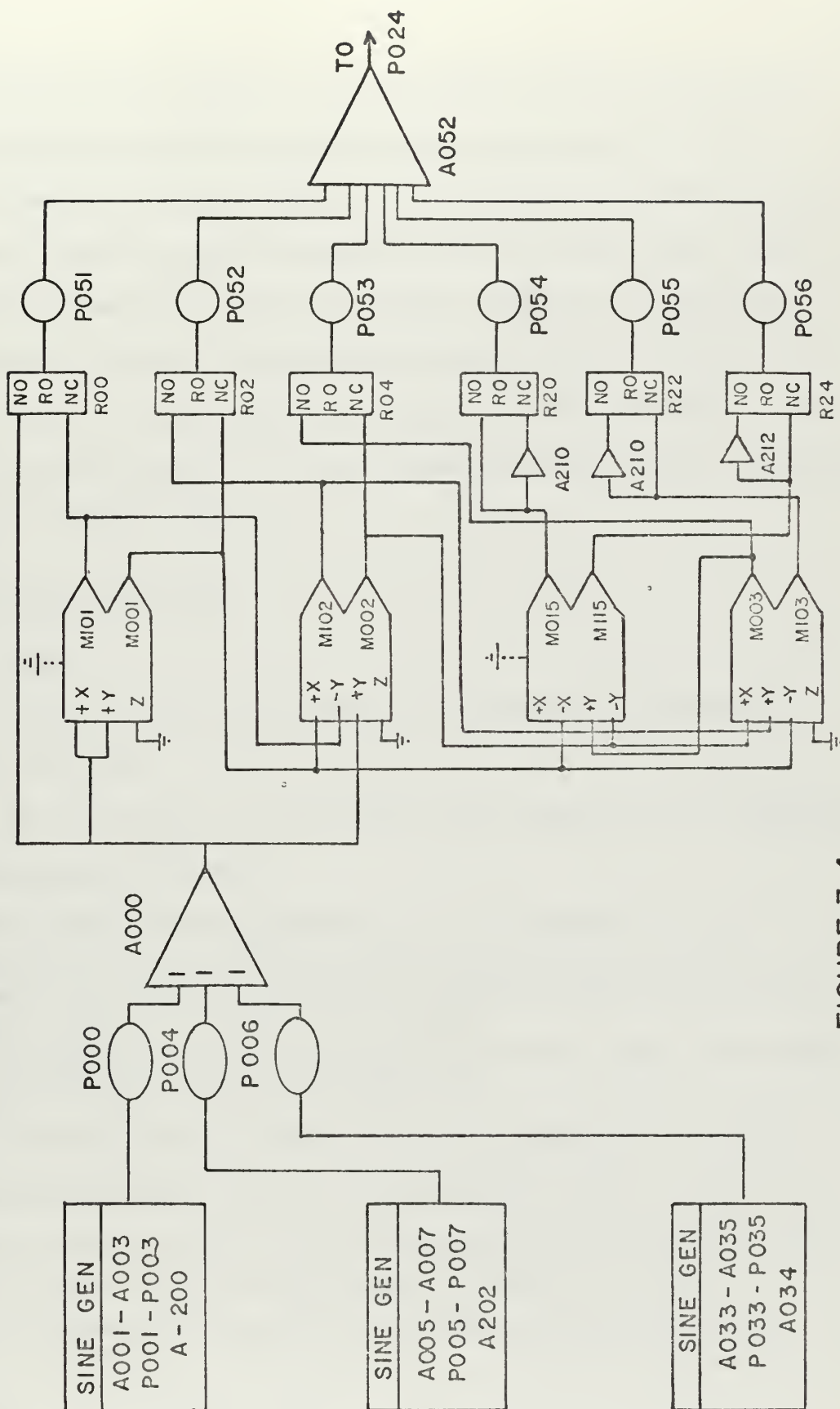


FIGURE 3.4

The parameters K are adjusted in order by P051 to P056 with the first three feeding the amplifier at $I = 10$ and the last at $I = 1$. This requires that P051 through P053 have a maximum of 10 while the sum of all three, times 10, plus the remaining three be 10000 or less. Any one of the parameters may have its polarity reversed by the switches R00, R02, R04, R20, R22, R24 activated by the lines T001 through T006.

Table II in the Appendix gives some of the combinations of coefficients and frequencies up to sixth harmonic when one or two frequencies constitute the input signal.

C. WHITE NOISE GENERATOR

1. Theory

To build a source of random noise a pseudo-random sequence generator was used.

As is well-known, a true random binary sequence may be classified by three properties [Ref.2]:

- a) The number of ones approximately equal to number of zeros;
- b) About half of the no-change states have the length of one, about one-fourth have length two, one-eighth have length three, and in general $1/2^n$ have the length n ;
- c) The expected value of the autocorrelation has a peak at the origin and decreases rapidly away from there.

A linear maximal sequence, like those generated by specially designed feedback shift registers, has properties very close to these:

- a) The number of ones is exactly one more than the number of zeros;
- b) The number of the no-change states of length n is $1/2^n$;
- c) The finite autocorrelation, the sequence considered periodic, has the form of

$$P(k) = \frac{1}{L} \sum_{n=1}^L a_n a_{n+k}$$
, where L is the sequence length and a_n the value of the n^{th} point of the sequence. This expression has only two values, say L when $k = 0, \pm L, \pm 2L, \dots$ and -1 otherwise.

Thus, within appropriate limits, a pseudo-random sequence may be used to simulate a true random one.

To generate binary analog pseudo-random noise from a random sequence, two techniques can be used. In one of them all bits are added modulo 2 for each shift. For the other only one bit is taken in each shift. This logic signal is then amplified and clipped to originate a pseudo-random square wave. The second technique was adopted.

The analog noise generated has first and second order statistics essentially identical to those of true random noise if the time of observation is short compared with the sequence period. Also, it is found in the literature that, after lowpass filtering, high confidence level tests indicate essentially a first-order Gaussian behavior [Ref. 7].

Again, for a long sequence and a relatively short observation interval, it is not possible to resolve the spectral lines that necessarily exist in the power spectrum of a periodic waveform.

2. Implementation

As shown in Fig. 3.5, a 25-bit feedback shift register (FSR) was constructed with bits 3 and 25 being added modulo 2 by the quarter-adder formed by A400, A200, A100, and then fed back to the input. This arrangement generates a maximal sequence with length $L = 2^{25} - 1 = 33, 554, 431$.

The output of the FSR and its inverse were used to actuate digital-analog switches, DA01 and DA02, which apply -100v or +100v, respectively, to the amplifiers A010 and A016, whose outputs are further combined in A014.

In order to avoid the all-zero state, bit one is set at reset time and all the remaining reset.

One requirement of a system built for signal analysis is the possibility to return to any observed situation under operator control.

[illegible]

FIGURE 3.5

For this reason, the shift pulse and the reset lines were made logically controlled by a special circuit, to be described later, so that the FSR operation is synchronized to the generator modes.

D. SIGNAL COMBINER

All three types of signals just described were brought to an amplifier, A024, through the potentiometers, P022, P023, P024, as shown in Fig. 3.6. The amount of injection of each one can be controlled and any combination achieved.

Potentiometer P015 was included to feed an initial condition to A015, if desired. With this feature and no inputs to A015 and A024, it is possible to simulate an impulse to the filters that follow and then analyze their characteristics.

Line T500 was connected at the output of the combiner to make it possible to sample before the input filter if desired.

E. BANDPASS FILTER

Actually, two filters were included after the combiner to simulate the transducer and the input circuitry. The original intention was to have a narrow-bandpass filter for the active detection simulation and the lowpass filter for the propeller signature problem.

1. Narrow-band Filter

A second-order filter, shown in Fig. 3.7 was installed for this purpose. Its frequency response is given by

$$= \frac{j\omega A}{(\omega_0^2 - \omega^2) + j\omega B}$$

where A is the gain and B the bandwidth related to the Q by $B \approx \frac{\omega_0}{Q}$.

SIGNAL COMBINER

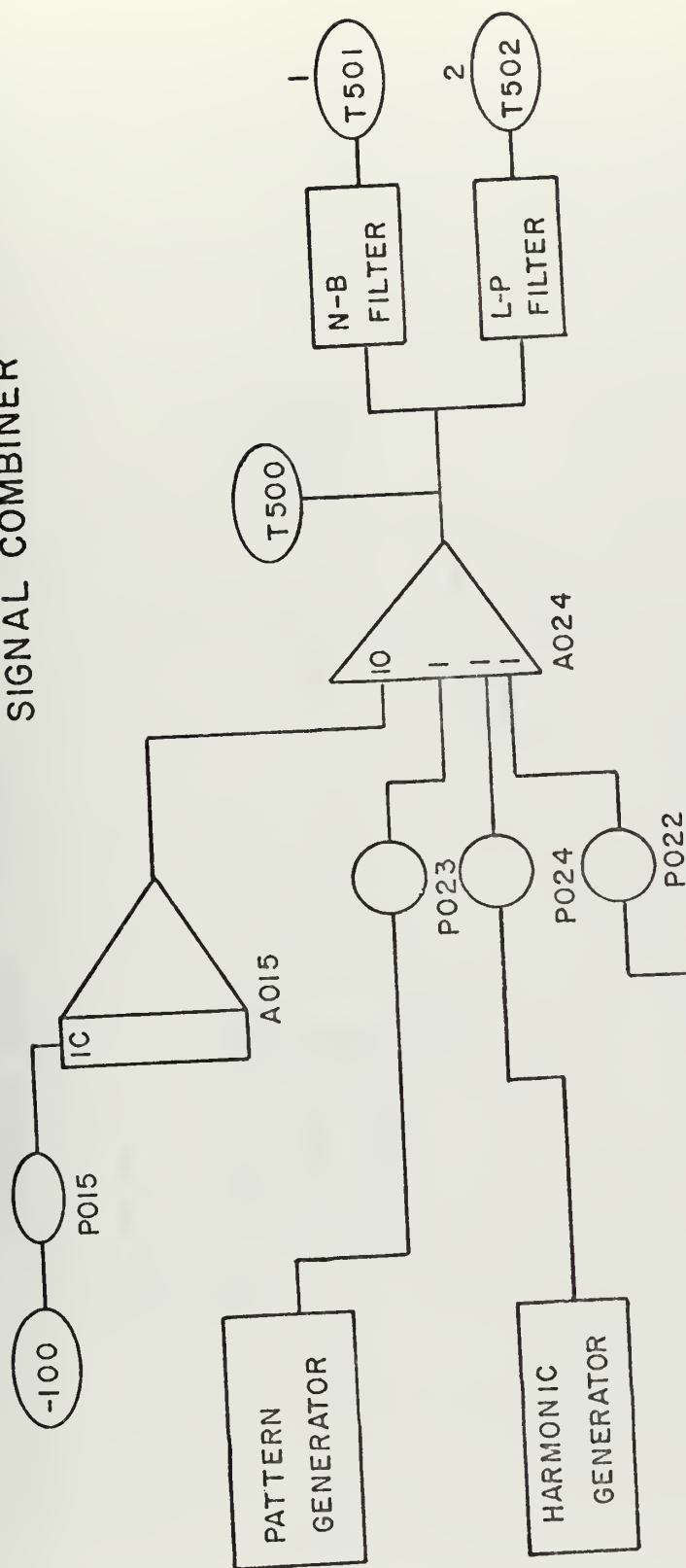
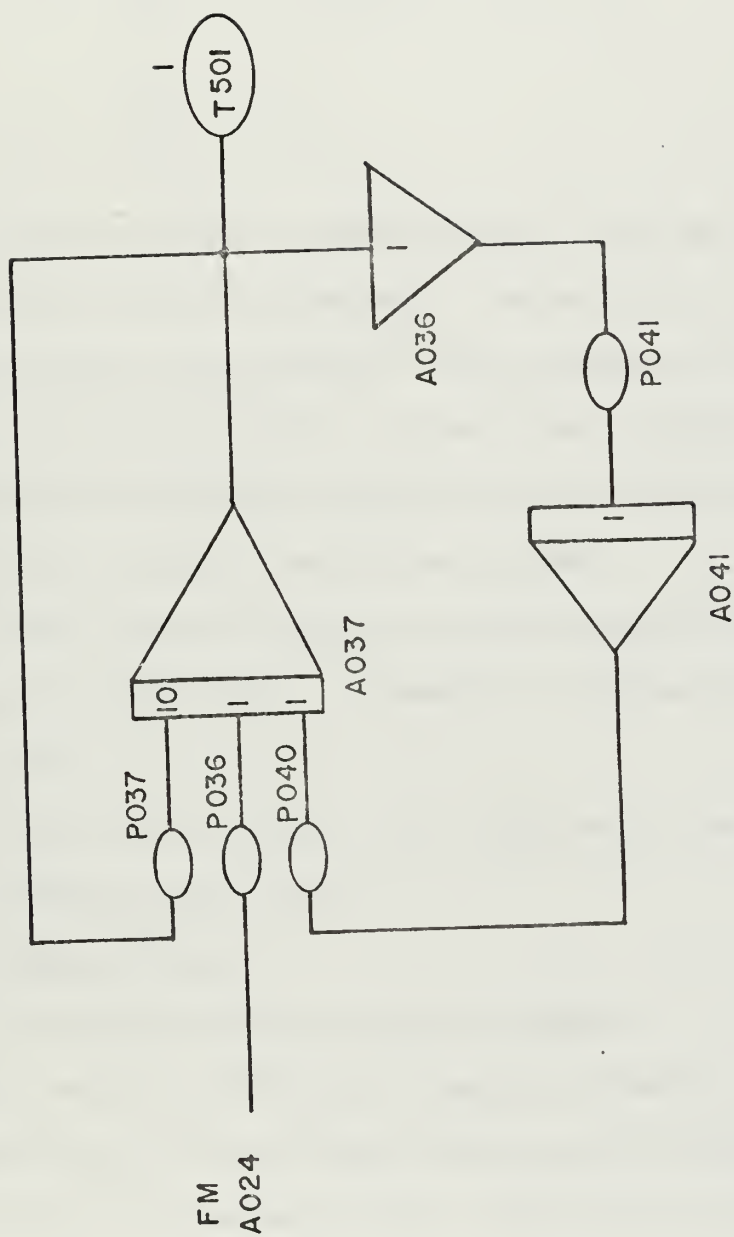


FIGURE 3.6

NARROW BAND FILTER



P036 - GAIN
P037 - BANDWIDTH
P040 - FREQUENCY
P041 - FREQUENCY

FIGURE 3.7

The differential equation describing this circuit is:

$$\ddot{y} + A\dot{y} + A\omega_0^2 y = -A\dot{x}$$

which, when integrated and put in terms of computer variables, becomes:

$$\overline{y} = -\int \frac{B}{\beta} \overline{y} dt_c - \int \frac{\omega_0}{\beta} \overline{z} dt_c - \int \frac{1}{\beta} x dt_c$$

$$\overline{z} = -\int \frac{\omega_0}{\beta} (-\overline{y}) dt_c$$

P036 then serves to adjust the gain, P037 the bandwidth, and P040 and P041 the central frequency. The values of Table I may also be used for these potentiometers if the frequency is substituted by $\beta A/2\pi$, Δf and f_0 respectively. To avoid small settings for P036 this potentiometer was normally connected to an amplifier having a gain ten times smaller than that used for the f_0 input.

The filter characteristic may be estimated by application of an impulse, as just mentioned. An example of this was taken and is shown in Fig. 1.9.

Line T501, connected at the filter output, makes it possible to take samples at this point.

2. Lowpass Filter

A simple RC filter was used for lowpass.

To assure the first-order Gaussian characteristic, the time constant RC has to be much greater than one shift period, Δt , but still less than $25\Delta t$, the sequence period. Of course, the inverse of RC is the cutoff filter frequency.

Table III in the Appendix was constructed to facilitate the adjustment of R, C, and Δt or the clock pulse frequency. To extend the

range of possible clock frequencies and make it easier to meet problem conditions, the built-in computer clock was passed through a frequency divider with outputs equal to one, one-half, one-fourth and one-eighth times the input.

If the narrow-band filter is in use, the shift pulse is set to make Δt much smaller, and $25\Delta t$ greater, than the inverse of the upper frequency.

F. INTERRUPT GENERATOR

This was one part of the system treated with very close attention. Since the digital computer program may be destroyed by interrupts arriving at the wrong times, they had to be carefully inhibited except when they should be expected.

Figure 3.8 shows the interrupt generator. As a general rule, the pulse that causes the interrupt is placed at the input of a NAND gate that has also an input connected to a line that is kept off by the program except during the sequence when it is legal for the occurrence of that particular interrupt. In this situation the output of the gate, if a pulse arrives, actuates a delay-flop that effectively generates the interrupt. Delay-flops were used to assure a short, well-shaped pulse.

Another interrupt generator is the circuit formed by the lines T013 (set), T113 (test) and the switch to ground. In the subroutine FASTCOR the digital computer tests line T113 once in every recursion. If the operator closes the switch at the display console, the line is set low and the program, after reading this value, branches out of the recursion loop, which corresponds to a software generated interrupt.

INTERRUPT GENERATOR

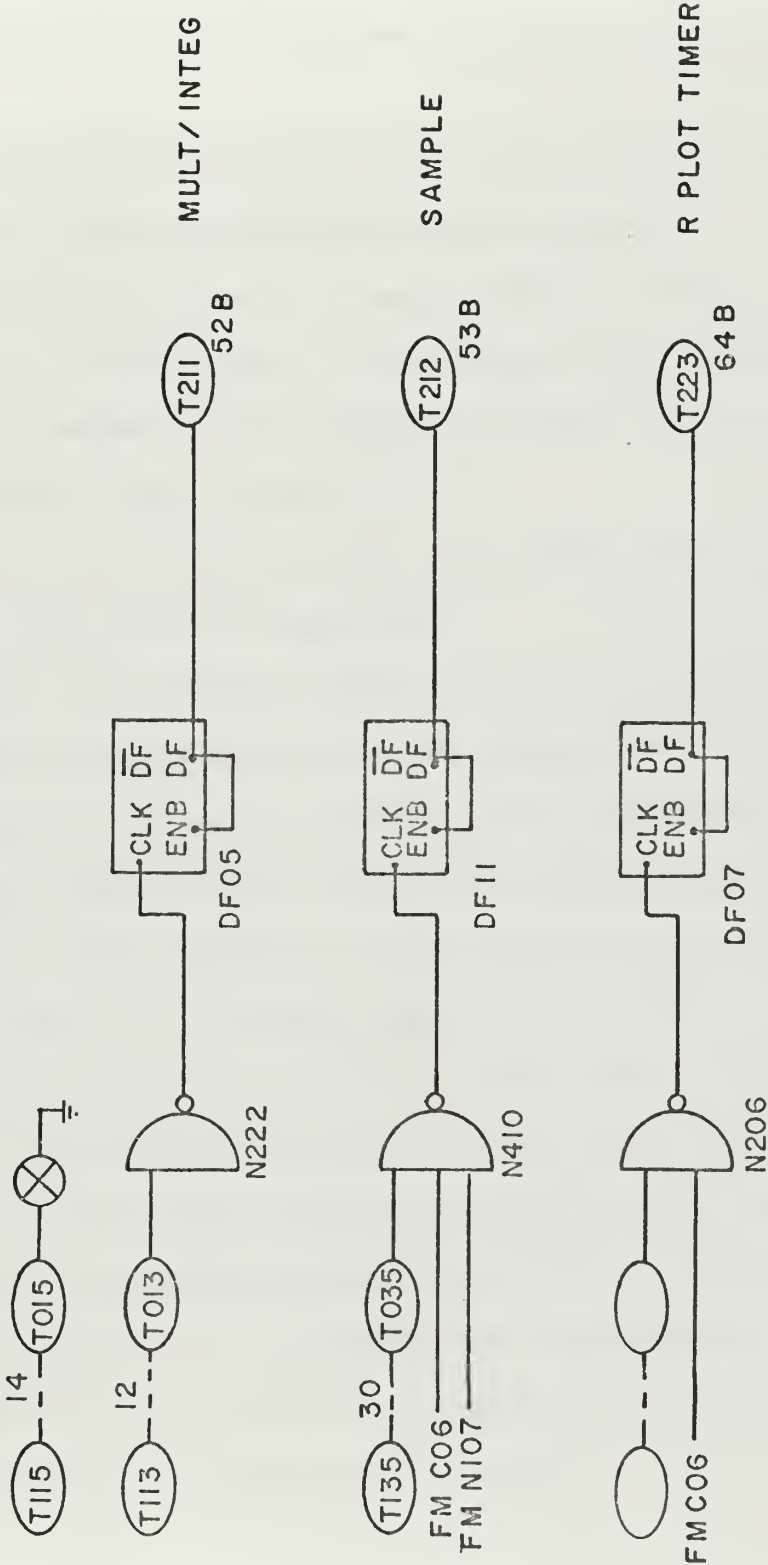


FIGURE 3.8

G. AUXILIARY CIRCUITS

In addition to the functional circuits described above, logic circuits were developed to insure the proper operation of the total system.

Among these, the synchronizer (Fig. 3.9) is responsible for keeping the noise generator in the same mode as the other generators. This circuit also prevents the occurrence of sample interrupts when the generators are not in a COMPUTE mode. In the Appendix are included the logic equations represented by the synchronizer. Other circuits and connections are shown in Figs. 3.10 and 3.11.

The real-time/time-scale controls were placed under digital command through the lines T030 and T031, respectively.

A four-bit counter was patched to work as a divider for the built-in clock to facilitate the rate selection for the feedback-shift register.

As a sample clock, a frequency synthesizer with square wave output was used. An analog comparator fed by this square wave generates a pulse starting at the zero crossing to trigger the sample interrupt, or to function as a timer for the recorder copy function, RPL0T.

The eight-channel recorder was connected to the outputs of the signal combiner, two of the sine generators and the pattern generator. Channels 6, 7, and 8 receive the copied curve through the DAC lines. Channel 5 displays a pulse whenever the third curve being plotted reaches the threshold setting; this pulse is generated by the probability counter.

The last circuit built was a multiplier-integrator for secondary computation of correlations by the subroutine ANACOR.

MODE SYNCHRONIZER

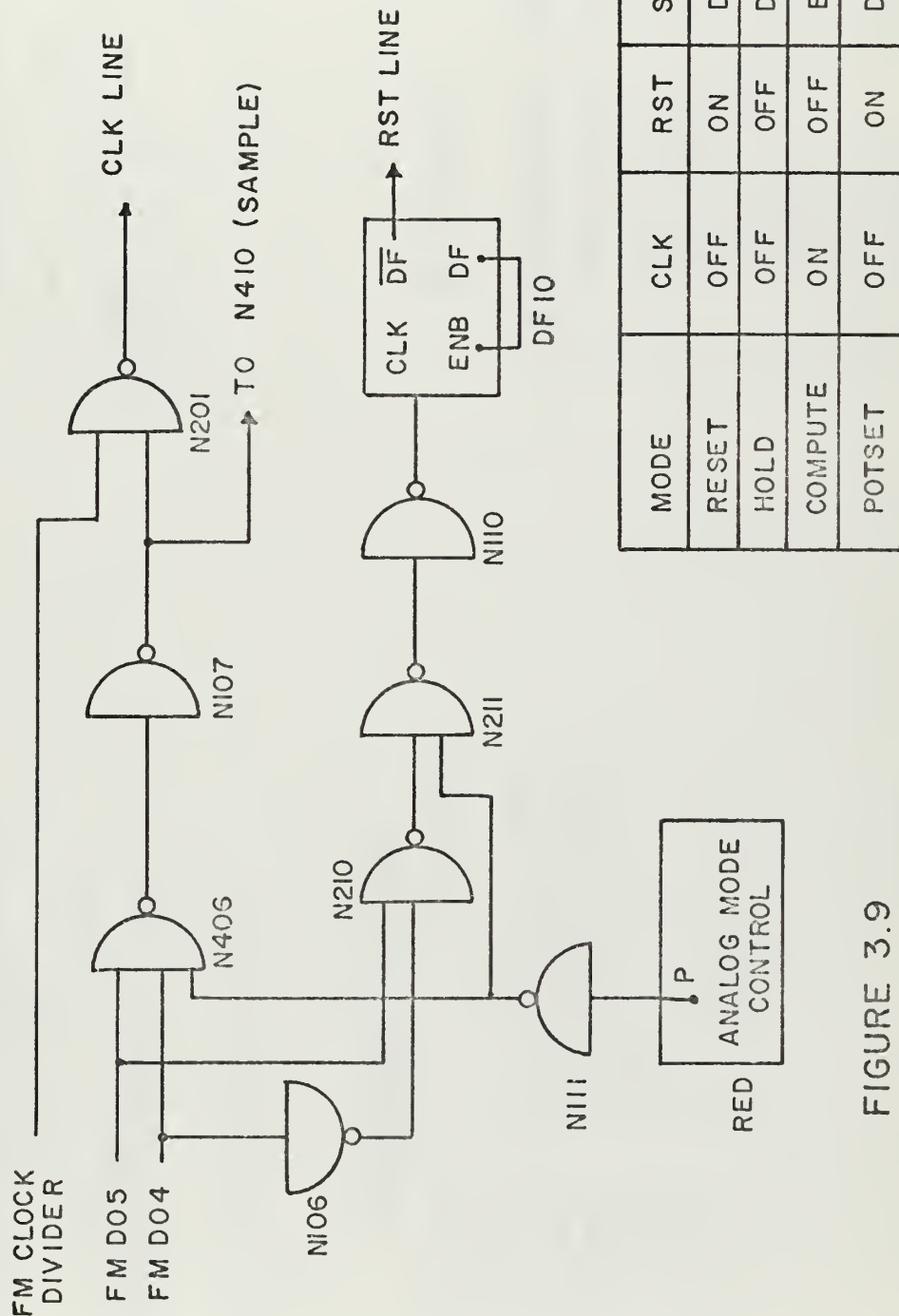
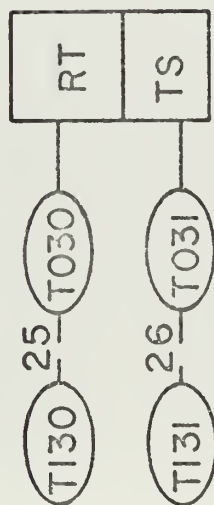


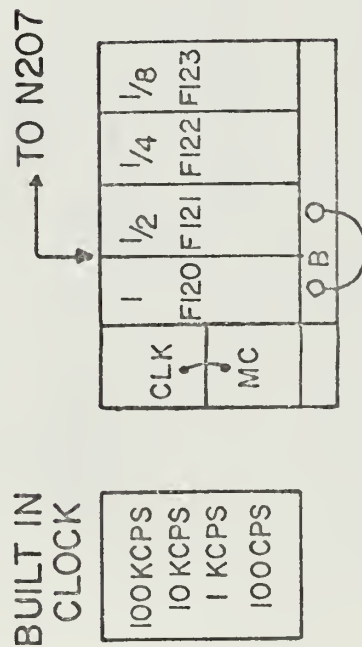
FIGURE 3.9

MODE	CLK	RST	SAMPLE
RESET	OFF	ON	DISAB
HOLD	OFF	OFF	DISAB
COMPUTE	ON	OFF	ENAB
POTSET	OFF	ON	DISAB

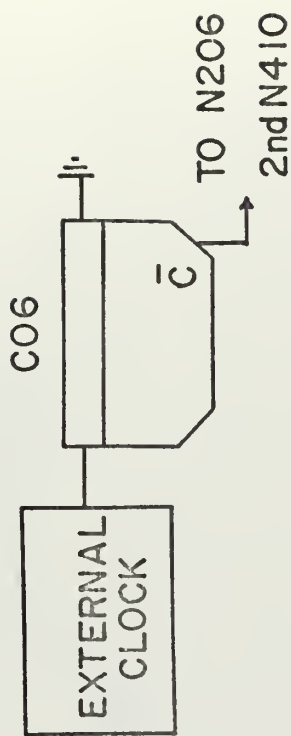
REAL TIME /TIME SCALE CONTROL



FREQUENCY DIVIDER



SAMPLE CLOCK

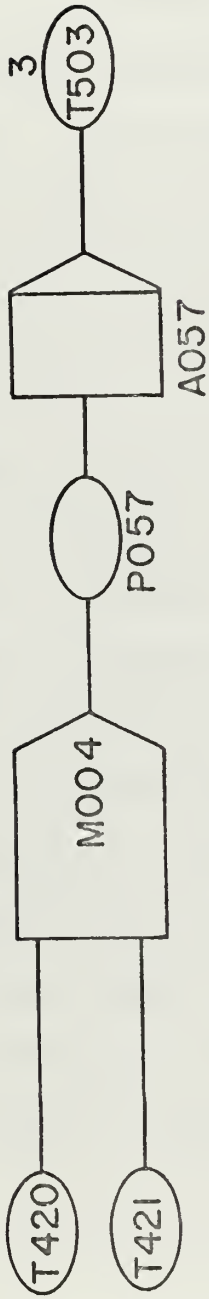


RECORDER

- 1 - COMBINER - A024
- 2 - SINE GEN - A003
- 3 - SINE GEN - A007
- 4 - PATT GEN - A017
- 5 - PROB COUNT-A044
- 6 - CURVE 1 - T425
- 7 - CURVE 2 - T426
- 8 - CURVE 3 - T427

FIGURE 3.10

MULTIPLIER INTEGRATOR



PROBABILITY COUNTER

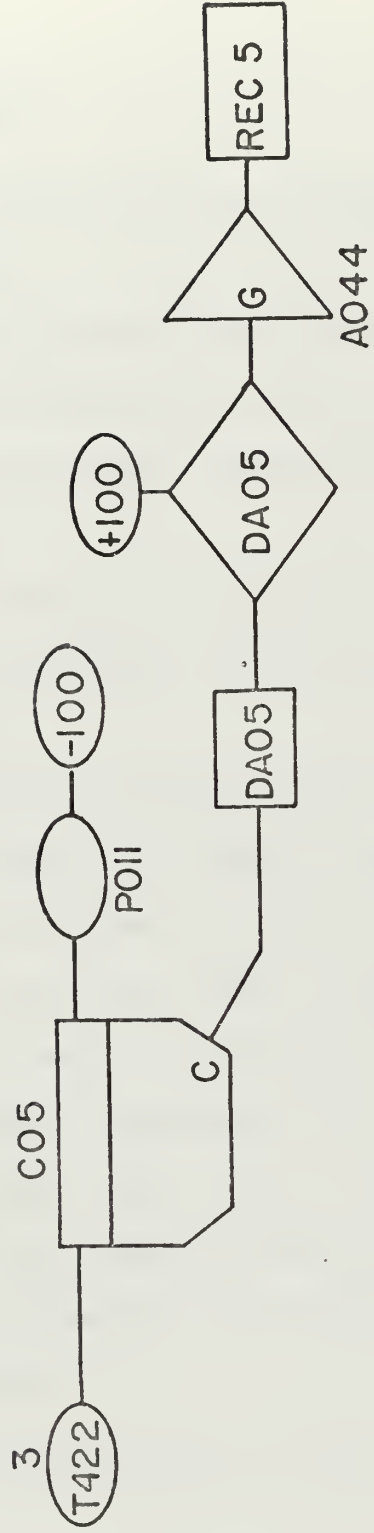


FIGURE 3.11

IV. GRAPHICS TERMINAL

The graphics computer provides the system one of its most important features, that of human-machine interaction.

The real-time operation enhanced by the natural form of graphical presentation made it easy for the operator to assimilate information, make decisions, and select the next sequence using the system control capability provided by the graphics console.

Since the AGT-10 is a general purpose computer it may be used to alter either the data or the form of the presentation to optimize the man-machine interface.

The system was controlled from the graphics terminal by typing simple messages on the screen using the console keyboard. A number of control possibilities are available. The operator could select to put the analog computer in any mode, or test its line values, select signals, sample, take the Fourier transform, perform auto- or cross-correlation and convolution, or determine power spectrum. He could also either initiate sequences to evaluate processing gain, ROC, distance, or signal-to-noise ratio or he may manually control every step of these operations. Finally, he had a choice to copy what was presented on the screen either on the line printer or on the paper recorder.

In the Appendix is given a more detailed description of the operating procedures.

V. SONAR SYSTEM APPLICATIONS

As a verification of the system capabilities several problems have been run to compare performances of the three processors, check the effects of integration time and bandwidth, and to understand the significance of information and joint information bandpass.

A. PRELIMINARY CONSIDERATIONS¹

To reproduce a signal from its sampled representation, in an ideal case, it is necessary to take samples at least at the Nyquist rate of two times the highest frequency contained in the signal. In the case of a narrow-band-limited signal, when the bandwidth is much smaller than the highest frequency, the rate becomes twice the bandwidth. The recovery of the signal, however, requires filtering with steep filters not found in practice.

For the system described, the only filtering action to which the samples are submitted is represented by the manner used to generate the vectors for display. This action can be compared with that of a first-order interpolator. For this reason, whenever the shape of the functions was desired, a sampling rate was used of several times the highest frequency, even in the case of bandpass limited signals since no bandpass filter was implemented.

For the time-limited and bandpass-limited functions the number of independent samples (n) is twice the time-bandwidth (BT) product as $n=2BT$.

¹Most of the expressions mentioned in this section are from, or follow directly from, those presented in the theoretical treatment of this subject by A. A. Gerlach, The Theory and Applications of Statistical Wave-Period Processing, Cook Electric, 1967.

At least this value is required to exactly represent the function and half of that if only the energy content is the object of the calculations. When computing statistical measures of sampled functions these numbers must be taken into consideration; in particular, due to the relationship between energy and autocorrelation, it is expected that the number of independent samples be $m = BT$ for this case.

The practical limitation of the bandwidth B requires that when the signal is processed no noticeable differences result if the bandwidth is extended beyond that limit. This of course depends on the capacity of the processor to "use" or "recognize" the information conveyed. In the case of a matched filter or linear correlator this optimum bandwidth approaches the information bandpass W_s [Ref. 4] where:

$$W_s = \frac{\left[\int_{-\infty}^{\infty} G(f) df \right]^2}{\int_{-\infty}^{\infty} G(f)^2 df}.$$

In the general case of detection, not only the desired signal is present, but also the undesired noise, and the purpose is then to identify signal in the background of noise. The effect of noise is to destroy part of the identifiable information or reduce the information bandwidth. This action is taken into account when the significant bandwidth is taken as the joint information bandpass or

$$W_{s,n} = \frac{\int_{-\infty}^{\infty} G(f) df \int_{-\infty}^{\infty} N(f) df}{\int_{-\infty}^{\infty} G(f) N(f) df}$$

From Shannon's expression for the information content, i.e.,

$$m = BT \log_2 (1+SNR) = [B \log_2 (1+SNR)] T$$

this interpretation can clearly be verified. Therefore, the information content of a signal in the presence of noise as seen by a cross-correlator type detector may be taken as:

$m = WT$ where W is the joint information bandpass for that particular processor, signal and noise, and the total number of independent samples $n = 2WT$.

It has been determined that for both the linear and the amplitude-limited cross-correlator, $W = W_{s,n}$; and for the clipper-limited processor $W = 1.5 W_{s,n}$. In any case the computation of $W_{s,n}$ is made with the signals as they are actually presented to the correlator, that is, undistorted, limited and clipped respectively.

B. PERFORMANCE COMPARISON

To compare the performance of the three processors, an FM slide signal was chosen with the following characteristics:

central frequency $f_0 = 5000$ Hz

FM-modulation — linear excursion of 500 Hz centered at f_0

sampling frequency — $f_s = 1000$ Hz

number of samples — $N = 256$

pulse duration or integration time — $T = N/f_s = 256$ msec

Since W was unknown the number of samples (N) could be greater, equal or smaller than the number of independent samples (n). However, considering that the signal power spectrum is entirely contained in the noise spectrum and assuming the noise-spectral-density constant over the physical bandwidth B , $W = B$ for the linear case and is greater for the nonlinear case. Thus,

$$n = 2BT = 2 \times 500 \times 256 = 256$$

This was then regarded as the linear processor having all the independent samples it was able to process, while the other two had less information than they were capable of processing.

Analysis of the results showed that the linear cross-correlator is always better than the nonlinear processors but it is necessary to consider that the last two were processing less information than they could for the preset bandwidth.

Also, it was observed that for small signal-to-noise ratio the performance of the nonlinear processors approached the linear processor. This is explained by inspection of Shannon's expression. As SNR increases, the nonlinear processors are unable to process the amount of information available since the amplitude limiting allows only two identifiable states. This process imposes an upper bound of $2B$ bits of information as seen by the processor. In other terms, since

$$SNR_o = f(p,w)SNR_i, \quad \text{then}$$

$$\Delta SNR_o = \frac{\Delta f(p,w)}{\Delta w} \Delta w SNR_i + f(p,w) \Delta SNR_i$$

For the nonlinear processors, when W increases $f(p,w)$ approaches $f(p) = \text{const}$ and $\Delta SNR_o = \Delta SNR_i \times \text{const}$

while with the linear correlator $f(p,w)$ increases with W .

At the lower end, if N is such that the full information may be processed by the nonlinear correlators, the wider bandwidth enhances their performance and no gain is achieved by the linear processor, since the extra samples are redundant or beyond its processing capability.

C. INTEGRATION TIME X BANDWIDTH

In this test the same FM signal was used with a reduced integration time.

To decrease the pulse duration, the number of samples was reduced to $N = 128$ while the sampling frequency was unchanged. Then T became 128 ms and the information content was cut to half of the previous value.

Only the clipper-limited processor was used. Comparison of these results with the others previously obtained revealed that the performance, or processor, gain as well as the detection sensitivity was reduced as expected.

A similar result could be achieved if the pulse duration was kept unchanged and the bandwidth divided by two since in this case the information content would also be halved.

D. INFORMATION BANDWIDTH

In order to explore further the meaning of the information bandwidth, a CW burst code at the frequency of 5000 Hz was generated. The number of samples was again 256 with a sampling frequency of 10 kHz with a resultant burst of 25.6 msec. The bandwidth of this pulse was approximately 40 Hz. The input filter was kept adjusted for a bandpass of 400 Hz.

A second run was also made with a sampling frequency of 100 kHz, reducing T to 2.56 msec and increasing B to 400 Hz.

The conclusion derived from the analysis shows that in the first case the processor "gain" was much higher than in the second. However, the ROC curves show that this contributed nothing to the detection sensitivity, but, on the contrary, decreased performance as is shown by the increase of false alarm rate. The reason for this rests on the fact that the bandwidth used was much higher than the information bandpass, and therefore unnecessary noise power was admitted over a band where the signal presented no significant energy.

This confirms the concept that processor gain is a significant parameter only if the bandwidth utilized corresponds to that compatible to the processor, and to signal and noise characteristics.

VI. CONCLUSION AND RECOMMENDATIONS

The final result of the study revealed the system as a powerful instrument for signal analysis. It can be used as an aid to design work or as a complement for theoretical studies in signal processing and detection.

The most important feature achieved, out of those initially specified, was that of the interaction between man and computer. Although still limited to some procedures, the operator can develop freely his reasoning in a successive stream of requests. Signal analysis can then be performed in a conversational and profitable manner.

In order to have this capability, the flexibility of the general purpose computers was efficiently explored. And no limits exist in this respect. Many more situations can be idealized and simulated, and other applications devised.

One field that immediately deserves closer attention is that of actual signal analysis. The system can handle this task without difficulty; however, only a continued experience can show which improvements are to be made. Depending on the nature of the problem it may be advisable to make some changes in order to trade flexibility and versatility for speed and precision. In this case, for instance, simplified versions of the subroutines FASTCOR, ANALYSIS and PROCESS could be used without overlay and with higher sampling rates to obtain very good statistical averages in a reasonable time.

Some other applications the writer had in mind to verify time permitting, are commented upon next. They are a suggestion for future studies with the system.

Input filter: More elaborate filters either analog or digital may be simulated to represent a transducer accurately.

Transmission medium: The effect of the transmission medium on the wave propagation may be well represented when the characteristics are known. For instance, the frequency selective transmission loss may be represented by a 20 db per decade type filter and the temperature and salinity influence on the sound velocity could be implemented based on the equations giving $c = f(T, T^2, S)$ [Ref. 9].

Multipath effects can be analyzed by using as a signal a combination of samples generated according to a predetermined delay pattern.

Doppler: A very simple way to simulate doppler shift situations is to detune the pattern generator during the replica sampling periods.

Statistical Wave Period Processors: This processor simulation can be obtained by a modification in the subroutine FASTCOR to count zero crossings of the cross-correlation function and then compute statistics upon this data.

Filter analysis: An example of this possibility was worked with the narrow band filter. Other filters either analog or digital may be analyzed with no difficulty. If a simple program is added, it would be possible to sketch desired filter waveforms on the screen and then analyze them.

Propeller signature recognition: To treat this very exciting problem the harmonic generator may simulate a signal with a well-defined line spectrum. It is not however an easy signal to detect when it is buried in noise, with low signal-to-noise ratios. The recursive mode of the correlation operation is appropriate to perform this computation without occupying excessive storage. This should be a good starting point to develop a special processor to handle this situation.

In conclusion, it is felt that the signal processing system presented here can be an effective tool for the exploration of processing methods in both instruction and research.

APPENDIX A

SYSTEM OPERATION

This Appendix includes a description of how to control the system from the graphics terminal.

Before loading the program, it is required to have accomplished the items below:

- a) Assign a unit to file 3.
- b) Connect the external clock (synthesizer) to C06 in the analog board and start the built-in clock.
- c) Connect the long cable switch between the upper leftmost tie point in the logic board and ground.
- d) Load and execute GATED in the AGT.

After loading, the teletype prints a reminder to have the analog ready, AGT GATED and clock running and asks for ID, the terminal identification number. When the message ID=1(or 2) is entered, the operation starts.

At the terminal only the keyboard is necessary to type the commands.

In the beginning, the constants are displayed and can be modified by erasing the block and typing the new value with the first digit aligned with the first letter of the heading. After carriage return the SETPOT mode is entered to initialize potentiometers. Then, the system waits for the first command.

Whenever an illegal operation or value is typed, the message "Incorrect Selection, Try Again" is displayed for a few seconds.

The commands are as follows:

A. CONSTANTS

M - Sample block size; initial 256; may be any power of two less than or equal to 256 and greater than 8.

- SAMP - 1, 2, or 3 indicates to sample replica, signal initial, or signal continuation; initialized with 1.
- THRS - threshold setting; 1000 means 1.0 normalized threshold; greater values are illegal; initialized with 0.
- LINE - line number of the sampling point; may be any ADK legal calling number; initialized with 1.
- PROC - type of processor; 0 is linear processor, 4 is clipper limited and 8 is amplitude limited; initialized with 0.
- POTS - initial value of pattern generator injection potentiometer; initialized with 9999; the noise generator potentiometer value is the complement to 10000; in a manual operation needs to be set only if the required sequence is correlation or convolution recursive.
- STEP - value of decreasing step for signal injection potentiometer to be used in performance or ROC determination; must be less than POTS; initialized to 4000.
- CPDT - if 5 indicates that false target rate is required; if 4 initiates a distance determination; if 3 a probability of detection; these operations, however take place only if a correlation recursive is asked for next; initialized to one.
- LNTH - length parameter for LPLLOT; has to be an even number between 4 and 16; initialized to 16 (smaller plot size).

All these are four-digit numbers and modifiable. The next three are displayed only.

SPOW - input signal power; initial 1.0

NPOW - input noise power; initial 0

SNR (DB) - signal to noise ratio in db, initialized to 0.

B. ANALOG CONTROL

SETPOT - displays alternately the words "POT VAL"; allowable potentiometer identifiers and values may be typed left justified under the first letter; sets the potentiometers.

SETLINES - displays alternately "SET VAL"; allowable line identifiers (two digits) and 1 or -1 may be typed left justified; sets the lines.

TEST - displays alternately "TST VAL"; allowable line identifiers may be typed; after return displays 1 or -1 according to the test; requires another return.

POTSET, COMPUTE, HOLD - puts the analog in the mode specified.

SCAN - displays current values of all amplifiers, potentiometers and several other active components, read in the current analog mode; waits for the message RETN to return or POTS to scan the potentiometers in POTSET mode; if the last option was entered only the potentiometer values are displayed and RETN must be typed to return.

C. ANALYSIS MODE

SAMP - asks for a confirmation about the constant values; if NO is typed, it brings in CONSTANTS; if return is keyed, executes sampling. (If a recursive correlation or convolution is to be performed, the replica must be sampled first with the potentiometer set to 9000 and next the signal, with the signal input potentiometer adjusted to the value read in POTS).

FFTR - asks for function to be transformed and displays function, absolute value and phase of the transform; results are not stored.

CORRELATION - asks for type and displays replica, signal and correlation; if recursive, the operation is terminated by the operator

closing the manual switch for a moment; also shows the threshold line and information about threshold reached, probability of detection (meaningful only if CPDT were previously set to 3 or 5), signal-to-noise ratio (the last one computed) and distances (if CPDT is set to 4).

CONVOLUTION - asks function to be convoluted with the replica, and type; displays replica flipped, the other function and convolution; if recursive, stopped by the manual switch.

POWER - asks for function and type (recursive not allows); displays autocorrelation and power spectrum (absolute value and phase); results are not stored.

AUTOCORRELATION - asks for function and displays function (twice) and result.

DISPLAY - asks for function and displays.

SNRI - external clock must be set for time scale operation; result may be obtained through CONSTANTS.

D. SIMULATION MODE

CWBURST - adjusts the circuits for a preset CW burst code

FMSLIDE - adjusts the circuits for a preset linear FM slide code.

FSK - sam for FSK code.

PROPELLER - same for a propeller signature; no automatic operations were implemented for this signal.

PERFORMANCE - computes the input-output characteristics for the processor and signal selected, and displays the result; external clock must be set for time-scale; an extra return required to end operation.

ROC - same as above but with the computation of Receiver Operating Characteristic

DIST - presets the circuits and initiates distance measurement operation.

F. PLOT

VPLOT - plots a VPLOT on the line printer.

LPLOT - plots a LONGPLOT on the line printer with dimension inversely proportional to LNTH.

RPLOT - asks to ready the recorder and informs the setting for the probability counter potentiometer if necessary; after return, informs that a delay of 20 seconds is in course to allow starting the eight-channel recorder; during this period a zero is sent to channels 6, 7 and 8 for calibration; the amplitude scaling is to be 2v per division and the speed 5 or 10 mm. per second.

G. STOP

Ends the program and releases the digital computer.

APPENDIX B

ANALOG COMPUTER INFORMATION

In this Appendix are derived the logic equations to control the PATTERN GENERATOR and the FSK SYNCHRONIZER. Also are included tables for quick scaling the potentiometers, (TABLE I), for determination of the coefficients of the frequencies generated by the HARMONIC GENERATOR (TABLE II) and for adjustment of the shift frequency of the FSR and the low pass RC filter (TABLE III).

The following variables are defined:

- RST - general reset mode
- CMP - general compute mode
- HLD - general hold mode
- RRP - Ramp reset mode
- CRP - ramp compute mode
- HRP - ramp hold mode
- R06 - switch R06 NC
- R10 - switch R10 NC
- R26 - switch R26 NC
- R25 - R input of A025
- H25 - H input of A025
- T07 - line T007 (program call 8)
- T10 - line T010 (program call 9)
- T11 - line T011 (program call 10)
- T15 - line T016 (program call 15)
- T17 - line T017 (program call 16)
- C04 - comparator C04
- C07 - comparator C07
- CWB - CW burst
- FM - FM slide burst
- FSK - FSK burst
- PRO - propeller simulation
- DIST - distance burst

PTS - potset
 SHC - clock line (FSR-shift)
 RLN - reset line (FSR)
 CDV - frequency divider
 SIN - sample interrupt enable

PATTERN GENERATOR

$$R06 = \overline{T17} = \text{DIST} + \overline{PR0}$$

$$R16 = \overline{T16} = \overline{\text{FSK}}$$

$$\text{RST} = \overline{T07} \cdot T10$$

$$\text{CMP} = \overline{T07} \cdot \overline{T10}$$

$$\text{HLD} = T07 \cdot \overline{T10}$$

The above equations are controlled by the program exclusively through CALL SETLINES statements.

$$R25 = \text{HRP} + \text{CRP}$$

$$H25 = \text{RRP} + \text{CRP}$$

$$\text{HRP} = \text{HLD} \cdot \overline{C07} \cdot \overline{CWB} = \overline{CWB} \cdot \overline{C07} \cdot T07 \cdot \overline{T10}$$

$$\text{CRP} = \text{CMP} \cdot \overline{C07} \cdot \overline{CWB} = \overline{CWB} \cdot \overline{C07} \cdot \overline{T07} \cdot \overline{T10}$$

$$\text{RRP} = \text{RST} + C07 + \overline{CWB} = \overline{CWB} + C07 + \overline{T07} \cdot \overline{T10}$$

$$\begin{aligned} R25 &= \overline{CWB} \cdot \overline{C07} \cdot \overline{T10} \cdot T07 + \overline{CWB} \cdot \overline{C07} \cdot \overline{T10} \cdot \overline{T07} \\ &= \overline{CWB} \cdot \overline{C07} \cdot \overline{T10} = \overline{\overline{\overline{\overline{\overline{CWB} \cdot \overline{C07} \cdot \overline{T10}}}}} \end{aligned}$$

$$\begin{aligned} H25 &= \overline{CWB} + C07 + \overline{T07} \cdot T10 + \overline{CWB} \cdot \overline{C07} \cdot \overline{T07} \cdot \overline{T10} \\ &= \overline{CWB} + C07 + \overline{T07} \end{aligned}$$

However C07 is instantaneously

$$\begin{aligned} C07 &= R25 \cdot H25 = \overline{CWB} \cdot \overline{C07} \cdot \overline{T10} \cdot (\overline{CWB} + C07 + \overline{T07}) \\ &= \overline{CWB} \cdot \overline{C07} \cdot \overline{T07} \cdot \overline{T10} \end{aligned}$$

and then

$$\begin{aligned} H25 &= \overline{CWB} + \overline{CWB} \cdot \overline{C07} \cdot \overline{T07} \cdot \overline{T10} + \overline{T07} \\ &= \overline{CWB} + \overline{T07} = \overline{\overline{CWB} \cdot T07} \end{aligned}$$

Making CW = T11 also controlled by the program, the equations:

$R25 = \overline{T11} \cdot \overline{C07} \cdot \overline{T10}$; $H25 = \overline{T11} \cdot \overline{T07}$ and $R10 = C04$ were implemented in the logic board as shown in Fig. 3.3. To prevent possible occurrence of a race problem and oscillations a short duration latch was included.

SYNCHRONIZER

$$\begin{aligned} SHC &= \overline{CMP} + PTS + \overline{CDV} = \overline{\overline{T07} \cdot \overline{T10}} + PTS + \overline{CDV} \\ &= T07 + T10 + PTS + \overline{CDV} = \overline{\overline{\overline{T07} \cdot \overline{T10} \cdot PTS \cdot \overline{CDV}}} \\ SIN &= \overline{CMP} \cdot \overline{PTS} = \overline{\overline{\overline{T07} \cdot \overline{T10} \cdot PTS}} \\ RLN &= \overline{RST} \cdot \overline{POT} = \overline{\overline{\overline{T07} \cdot \overline{T10} \cdot POT}} = \overline{\overline{\overline{T07} \cdot \overline{T10} \cdot PTS}} \end{aligned}$$

The implementation is shown in Fig. 3.9.

TABLE I
POTENTIOMETER SETTING

Real time C = .1

Time scale C = .01

$$P = \frac{20000 \text{ fC}}{\beta I}$$

$$f_{sc} = \frac{f}{\beta}$$

$$F = \frac{f_{sc}}{I} = \frac{f}{I\beta}$$

F	P	F	P	F	P	F	P
1.59	9999	1.19	7480	.79	4960	.39	2445
1.57	9986	1.17	7350	.77	4840	.37	2325
1.55	8874	1.15	7230	.75	4715	.35	2198
1.53	8861	1.13	7200	.73	4585	.33	2055
1.51	9495	1.11	6970	.71	4455	.31	1948
1.49	9352	1.09	6840	.69	4340	.29	1821
1.47	9235	1.07	6720	.67	4205	.27	1695
1.45	9120	1.05	6595	.65	4085	.25	1570
1.43	8995	1.03	6465	.63	3948	.23	1443
1.41	8860	1.01	6345	.61	3835	.21	1319
1.39	8725	.99	6220	.59	3703	.19	1193
1.37	8610	.97	6095	.57	3580	.17	1058
1.35	8490	.95	5965	.55	3478	.15	0942
1.33	8351	.93	5840	.53	3325	.13	0816
1.31	8230	.91	5720	.51	3205	.11	0691
1.29	8102	.89	5590	.49	3078	.09	0565
1.27	7970	.87	5460	.47	2935	.07	0439
1.25	7851	.85	5340	.45	2825	.05	0314
1.23	7730	.83	5215	.43	2700	.03	0188
1.21	7600	.81	5095	.41	2575	.01	0063

TABLE II
HARMONIC CONTENT

$$e = f(f_1)$$

Frequency	Coefficient
DC	$-K_2 a^2/2 + 3K_4 a^4/8 - 5K_6 a^6/16$
f	$K_1 a - 3K_3 a^3/4 + 5K_5 a^5/8$
$2f$	$K_2 a^2/2 - K_4 a^4/2 + 15K_6 a^6/32$
$3f$	$K_3 a^3/4 - K_5 a^5/16$
$4f$	$K_4 a^4/8 - 3K_6 a^6/16$
$6f$	$K_6 a^6/32$

$$e = f(f_1, f_2)$$

Order	Frequency	Amp. due to lowest K
1	K_1, K_3, K_5 f_1, f_2	$K_1 a, K_1 b$
2	K_2, K_4, K_6 $2f_1, 2f_2$ $f_1 \pm f_2$	$1/2 K_2 a^2$ $K_2 a$
3	K_3, K_5 $3f_1, 3f_2$ $2f_1 \pm f_2$	$1/4 K_3 a^3$ $3/4 K_3 a^2 b$
4	K_4, K_6 $4f_1, 4f_2$ $3f_1 \pm f_2$ $2f_1 \pm f_2$	$1/8 K_4 a^4$ $1/2 K_4 a^3 b$
5	K_5 $5f_1, 5f_2$ $4f_1 \pm f_2$ $3f_1 \pm 2f_2$	$1/16 K_5 a^5$ $5/16 K_5 a^4 b$
6	K_6 $6f_1, 6f_2$ $5f_1 \pm f_2$ $4f_1 \pm 2f_2$ $3f_1 \pm 3f_2$	$1/32 K_6 a^6$ $3/16 K_6 a^5 b$



TABLE III
FSR FREQUENCY AND RC FILTER

$$f_{cs} = \frac{f_{cut}}{\beta}$$

Built in Clock	Divider ÷	fcs	R	C	Series/Parallel
100	8	1.25	.1	2x10	S
				3x1	P
100	4	2.0	.1	2x10	S
100	2	3.33	.1	3x1	P
100	1	5	.1	2x1	P
1000	8	12.5	.1	2x1	S
				3x.1	P
1000	4	20	.1	2x1	S
1000	2	33.3	.1	3x.1	P
1000	1	50	.1	2x.1	P
10000	8	125		3x1	S
10000	4	200	.1	2x10	S

COMPUTER PROGRAM
FOR
HYBRID SONAR SIMULATION

MAIN PROGRAM

Purpose: The main program is used for computer initialization and serves as a bridge to keep the status during overlay swapping. It also creates two dummy pulses as Replica and Signal for operative checking after initialization.

Parameters and variables: In the explanation of the program the meaning of a variable is given only in the subroutine where it is effectively used. In the case of the MAIN program the variables below are of interest:

- M - basic sample size. Has to be a power of two. Initially has the value 256.
- YR - array for storage of Replica, within size M. Initially dimensioned to 256.
- YS - array for storage of signal, with size 3M. Initially dimensioned to 768.
- YC - array for storage of correlation, with size 2M. Dimensioned to 512.
- VAGO - buffer, with size 5M and dimensioned to 1280.
- IB - buffer, with size 2M plus one word. Dimensioned to 1025.
- MT - text directory.
- MG - graph directory.
- ID - graphics terminal number.


```

C0484N IJ,I,INDEX,IPLT,I,ILP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,0
C0484N L9G(3),LINE,MAC,ARRAY,RTH,IC9JNT,IMEM,PR9B
C0484N ISA,P,IDIIST,ISTA,ISTB,ISTC,ISTD,ISTE
C0484N ISTF,IKT,SPR(160)
CALL DECT
C0484N YS(768),YC(512),YR(256),VAG0(1280),IBL9CK(1025)
DIMENSION YS(8),YT(42)
C0484N DECT(42),C=V(YR,YS,M)
C0484N YCT(73,SAMPLE)
C=X;NT=42;N=255
I,I=1
LINE=1 DEY=1J=ISTC=ISTD=1
IPL9T=ITP=ISAMP=ISTB=0
ISTA=9999
ICTE=4270
ILE=16
C0484N X=SP9W=C,
C0484N C,P=RC9F=SC9F=3=1.
ISTF=10
,A,PL9T I,
C0484N I=1,6**
03 YS(I)=C,
C0484N I=51,100
51 YS(I)=YS(I+50)=YS(I+100)=YS(I+150)=Y3(I+150)=*3
TEIN) 3
WRITE(3)(YC(I),I=1,3*M),(YC(I),I=1,2*M),(YR(I),I=1,M)
OUTPUT(101) ANALOG READY -CLOCK RUNNING - AGT GATED ,
OUTPUT(101) ENTER ID,
I,OUT(101)
CALL INITIAL(1,111S,MG,NG,MT,NT,IBL9CK,M)
CALL INITIAL(IJ,105S,MG,NG,MT,NT,IBL9CK,M)
CALL INITIAL(2,105S,MG,NG,MT,NT,IBL9CK,M)
CALL INITIAL(3,106S,MG,NG,MT,NT,IBL9CK,M)
CALL DECT(1,IBL9CK,MT,P,10PS,102S,IBL9CK(200),IBL9CK(400))
CALL DECT(1,IBL9CK,MT,1,10PS,103S,IBL9CK(200),IBL9CK(400))

```



```

106 G3 I8 (111,...,118)I1
111 CALL CPGT(104S,102S,IBLOCK,IBIT,4,112S,MT)
112 CALL GNAPHAL(104S,102S,VAG,MT,101S)
113 CALL AAIYGIS(104S,102S,YS,YR,YC,VAG9,IBLOCK,MT,M,O)
114 CALL PRCFSS(104S,102S,YS,YR,YC,VAG9,IBLOCK,MT,M,O)
115 CALL DEFANAL(104S,102S,YS,YR,YC,VA59,IBLOCK,MT,M)
116 CALL PRLCT(M+M,YC,IBLOCK,102S,MT)
117 CALL CIRCPAT(102S)
118 STOP

```



```

SUBROUTINE SAMPLE
COMMON IJ,II,INDEX,IPLAT,ILP,ITR,ID,C0F,SC0F,RC0F,SNR,SP0W,NP0W,Q
COMMON LAG(3),LINE,MAC,ARRAY,RTH,ICOUNT,IMEM,PR0B
REAL NP0W
CALL ALK(LINE,ARRAY)
MAC=MAC+1
RETURN/END

```

Purpose: executes the sampling of the analog computer when connected by an interrupt.

Variables:

LINE - line to be sampled

ARRAY - storage for the sample

MAC - counter for the samples


```

COMMON IJ,II,INDEX,IPL61,ILP,ITR,ID,C0F,SCE0F,RC0F,SNR,SP0W,RP0W,Q
DIMENSION YA(1),YB(1)
L=II-IJ;K=''-IJ
CALL DAC(1,YA(L),2,YB(K))
IJ=IJ+1
*
RETURN ;END

```

Purpose: controls the transmission of values to analog multiplication and integration where the subroutine ANACOR is being used. Connected through interrupt.

Variables:

II,IJ - used to transfer information from and to ANACOR

YA,YB - values to be multiplied

SUBROUTINE INITIAL

Purpose: initializes text and graph or erases those already displayed. Sets several lines initially, mainly those that inhibit the interrupts.

Parameters and variables:

ITYPE - pointer to the option, i.e., to initialization, text or graph erasing.

LABEL - return address

ID,IG,MT - as in MAIN

NG,NT - graph and text directory size


```

SUBROUTINE INITIAL(ITYPE,LABEL,NG,NG,MT,NT,IB,M)
COMMON IU,II,INDIX,IPLST,IIP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,Q
COMMON LPO(3),LINE,MAC,ARRAY,RTH,IC9UNT,INEM,PROB
COMMON ISAMP,IDIGT,ISTA,ISTB,ISTC,ISTD,ISTF
COMMON ISTF
IF ENSTRT NG(NG).MT(MT),IB(NG)
REAL V9W
NM=M+1
GO TO(910,920,930,940)ITYPE
910 CALL DINIT(D,NG,NG,IE)
IF IE.EQ.0,OUTPUT(101)IE,'910-INITIAL-G'
CALL DINIT(D,MT,NT,IE)
IF IE.EQ.0,OUTPUT(101)IE,'910-INITIAL-T'
IF IYPE.EQ.4,GO TO 920
CALL SCIPRT(4REC15,0)
CALL SCILNFB(5,1,9,-1,10,1,12,1,16,1,25,1,26,-1,29,1,30,1)
CALL ENABLC
920 GO 911 I=1,24
941 IE(I)=77777777
942 GO 912 I=1,40
CALL IXT9(I,IB,1,1,1,1,3,IE)
IF IE.EQ.0,OUTPUT(101)IE,'941-INITIAL'
912 CONTINUE
ISTD=1;ISTF=10
IF IYPE.EQ.3,RETURN LABEL
930 GO 931 I=1,99
931 IE(I)=0
942 GO 932 I=1,17
IF I.GT.3,M=5;IF I.EQ.4,M=3
CALL GRAP9(I,IE,M,I,IF)
IF IE.EQ.0,OUTPUT(101)IE,'931-INITIAL'
932 CONTINUE
RETURN LABEL;END

```


SUBROUTINE DECO

Purpose: decodes the control codes typed and sets pointers to indicate the sequence to be followed; also displays the error message when invalid selections are entered.

Parameters and variables:

IB - working buffer

ID,MT,LABEL - defined as previously

MABEL - return address after an error message

IT - pointer to indicate decoding operation or error message display

LF - code table storage

II - pointer to the general selection

IJ - pointer to the specific selection


```

SURRJUINE OFC(I2,MT,IT,MABEL,LABEL,LF,I9P)
C=9,IJ,II,INDEX,IPLT,ILP,ITR,IO,COF,SCOF,RCOF,SNR,SPBW,NPBW,Q
REAL,NSBW
TYPE=512N LF(IT),J9P(IT),I2(IT),MT(IT)
IF IT.EQ.2,90 TO 904
ASSIGN 1 TO LF(1)
ASSIGN 11 TO LF(11)
ASSIGN 12 TO LF(12)
ASSIGN 13 TO LF(13)
ASSIGN 14 TO LF(14)
ASSIGN 15 TO LF(15)
ASSIGN 16 TO LF(16)
ASSIGN 17 TO LF(17)
ASSIGN 18 TO LF(18)
ASSIGN 21 TO LF(21)
ASSIGN 22 TO LF(22)
ASSIGN 23 TO LF(23)
ASSIGN 24 TO LF(24)
ASSIGN 25 TO LF(25)
ASSIGN 26 TO LF(26)
ASSIGN 27 TO LF(27)
ASSIGN 31 TO LF(31)
ASSIGN 32 TO LF(32)
ASSIGN 33 TO LF(33)
ASSIGN 34 TO LF(34)
ASSIGN 41 TO LF(41)
ASSIGN 42 TO LF(42)
ASSIGN 43 TO LF(43)
ASSIGN 44 TO LF(44)
ASSIGN 51 TO LF(51)
ASSIGN 52 TO LF(52)
ASSIGN 53 TO LF(53)
ASSIGN 61 TO LF(61)
ASSIGN 62 TO LF(62)
ASSIGN 63 TO LF(63)

```



```

AS3134 64 18 LF(64)
AS3135 71 18 LF(71)
REPEAT 900, FOR I=19,28,29,(35,39),(45,49),(54,59),(65,69)
900 AS3134 100 18 LF(I)
REPEAT 901, FOR I=1,(11,19),(21,29),(31,39),(41,49),(51,59),(61,69)
C,71
REPEAT(4,LF(I),18P(I))
901 I(I)=77777777
CALL EXPR(10,13,1,38,1,1,3,IE)
IF IE.L.C.INPUT(101)IE,DECODE-901'
902 IF 900(M(38),8).EQ.0,99 18 902
CALL EXPR(10,13,1,0,30,IE)
IF IE.E.O.INPUT(101)IE,DECODE-902'
REPEAT(4,901,13)18
REPEAT 903, FOR I=1,(11,19),(21,29),(31,39),(41,49),(51,59),(61,69)
C,71
IF I+10
903 IF IE(1).EQ.18P(I),99 18 906
904 905 I=1,8
905 I(I)=77777777
REPEAT(32,902,18)
CALL EXPR(10,13,8,39,1,2,3,IE)
IF IE.E.O.INPUT(101)DECODE-905'
LAY=2.18;CALL DELAY
RETURN 905L
906 IO=9(11,10);II=II/10
IF II.E.1,RETURN
IF II.E.6,IPLAT=IPLAT+1;RETURN
RETURN LABEL
907 REPEAT(A4)
908 REPEAT(INCORRECT SELECTION•TRY AGAIN•')
1 REPEAT(C9,9)
11 REPEAT(GETP)
12 REPEAT(SETIL)
13 REPEAT(TEST)

```



```

14 FORMAT('PTS')
15 FORMAT('COMP')
16 FORMAT('HOLD')
17 FORMAT('RESET')
18 FORMAT('SCAN')
19 FORMAT('SAFE')
20 FORMAT('FFTR')
21 FORMAT('C3R')
22 FORMAT('C3V')
23 FORMAT('POME')
24 FORMAT('DISP')
25 FORMAT('AUT')
31 FORMAT('SN1')
32 FORMAT('SE1')
33 FORMAT('RGC')
34 FORMAT('DIS1')
35 FORMAT('A1C')
36 FORMAT('A1G')
37 FORMAT('DISC')
38 FORMAT('REC')
39 FORMAT('VPL')
40 FORMAT('LPL')
41 FORMAT('PDL')
42 FORMAT('C1C')
43 FORMAT('F1C')
44 FORMAT('ESX')
45 FORMAT('PK1')
46 FORMAT('STEP')
47 FORMAT('C1C')

```

END

SUBROUTINE CONST

Purpose: displays some of the problem parameters and allows their modification under operator control. The modified parameters are accepted only if within preset limits.

Parameters and Variables:

ID, LABEL, IBLOCK, MT - as previously defined

MABEL - return address if an error occurs

NABEL - return address when the program is initialized

IBIT - flag to indicate if the subroutine is called for the first time after initialization

The following parameters are described in Appendix A under their calling name, according to this correspondence table:

M - M; INDEX - SAMP; ITR - THRS; LINE - LINE; ISAMP - PROC; ISTA - POTS;

ISTE - STEP; ISTC - CPDT; ILP - PLOT.


```

SUBROUTINE CONST(MABEL,LABEL,IBLOCK,IBIT,M,NABEL,MT)
COMMON IU,II,INDEX,IPLT,ILP,ITR,ID,C9F,SCDF,RCDF,SNR,SPBW,NPBW,Q
COMMON L23(3),LINE,MAC,ARRAY,RTH,ICJUNT,IMEM,PR8B
COMMON ISA'P,IOIST,ISIA,ISTB,ISTC,ISTD,ISTE
REAL NPBW
C INENSION IBLOCK(2),MT(2)
DOCODE(36,1,IBLOCK)
CALL TEXTA(IC,IBLOCK,24,29,1,1,3,IE)
IF IE.EC.GOUTPUT(101)IE,111
J=1;REPEAT 1101, FKR I=4,INDEX,ITR,LINE,ISAMP,ISIA,ISTE,ISTC,ILP
DOCODE(4,2,IBLOCK(J))I
IBLOCK(J+1)=4H
1101 J=J+P
J=18;IBLOCK(24)=4H
REPEAT 1102, FKR R=SP4,NPBW,SNR
DOCODE(8,2,IBLOCK(J))R
1102 J=J+8
CALL FIXTR(ID,IBLOCK,24,40,1,1,3,IE)
IF IE.EC.GOUTPUT(101)IE,1102
1103 IF IC(MT(40),8).EC.C.G.F 1103
CALL FIXTI(IC,IBLOCK,24,2,40,IE)
IF IE.L.C.GOUTPUT(101)IE,1103
DOCODE(4,2,IBLOCK(1))MY
IF NM.LI.8.53.N.GT.512, RETURN MABEL
V=0
DOCODE(4,2,IBLOCK(3))I'D
IF I'D.LI.1.0R.I'D.GT.3, RETURN MABEL
I'DFX=I'D
DOCODE(4,2,IBLOCK(5))JTR
IF JTR.LI.-999.9R.JTR.BT.1000, RETURN MABEL
I'D=JTR
DOCODE(4,2,IBLOCK(7))LIN
IF LIN.LI.1.0R.LIN.GT.31, RETURN MABEL
LI'E=L'I
DOCODE(4,2,IBLOCK(9))L'N

```



```

IF LIN.LT.0.9R.LIN.GT.9.9R.M9D(LIN,4).NE.0,RETURN MABEL
IS4=LIN
CECODE(4,2,13)LOCK(11))LIN
IF LIN.LT.0.9R.LIN.GT.9999,RETURN MABEL
161A=LIN;13TR=10000-LIN
CECODE(4,2,13)LOCK(13))LIN
IF LIN.GT.151A,RETURN MABEL
15TF=LIN
CECODE(4,2,13)LOCK(15))15TC
CECODE(4,2,13)LOCK(17))LIP
IF LIP.LT.4.9R.LIP.GT.16.9R.M9D(LIP,2).NE.0,RETURN MABEL
16=LI
IF 16IT.10.1,16IT=0;1J=1;RETURN MABEL
1 16ITAT(16)      S4=2      16RS      LINE      PRAC      P9TS      STEP      CP
CE=LNTH      SP9A      169W      SNR(05) ')
2 16ITAT(14)
3 16ITAT(17.0)
RETURN MABEL;END

```


SUBROUTINE CIRCPAT

Purpose: adjusts potentiometers and sets lines in order to configure four types of transmission codes.

Parameters and Variables:

LABEL - return address

IJ - pointer to indicate selection; 1 - CW; 2 - FM; 3 - FSK; 4 - PROPELLER

ISTA - signal injection potentiometer adjustment

ISTB - noise injection potentiometer adjustment


```

SU ROUTINE CJRCPAT(LABEL)
CJRM04 IU,II,INDEX,IPLCT,ILP,ITR,IO,C9F,SC9F,RC9F,SNR,SP9W,NP9W,0
CJRM04 LOG(3),LINE,MAC,APRAY,RTH,ICOUNT,IMEM,PROB
CJRM04 ISA'P,IDIIST,ISTA,ISTB,ISTC,ISTD,ISTE
CALL CJRCP
CALL SETPAT(4HP022,ISTB,4HP015,0,4HP017,400,4HP021,500)
IF IU.EQ.4,GO TO 400
CALL SETLINES(16,1)
CALL SETPAT(4HP023,ISTA,4HP024,0,4HP036,1000,4HP037,9600,4HP040,
    0,4HP041,9100)
IS IU.E.1,GO TO 200
CALL SETLINES(10,1,15,1)
CALL SETPAT(4HP026,9612)
RETURN LABEL
200 CALL SETLINES(10,-1,15,1)
CALL SETPAT(4HP025,1111,4HP026,9051,4HP027,9999)
IF IU.E.2,RETURN LABEL
CALL SETLINES(15,-1)
CALL SETPAT(4HP030,9051,4HP031,9999,4HP032,9612)
RETURN LABEL
400 CALL SETPAT(4HP023,0,4HP024,ISTA,4HP000,4000,4HP004,3000,4HP006,
    0,2000,4HP001,1584,4HP003,1584,4HP005,2514,4HP007,2514,4HP033,2199,
    0,4HP035,2109,4HP051,050,4HP052,0,4HP053,0,4HP054,2000,4HP055,5000,
    0,0,0,0)
CALL SETLINES(2,1,3,1,4,1,5,-1,6,1,7,-1)
RETURN LABEL:END

```

*

SUBROUTINE GRAPHAL

Purpose: Allows control of the analog computer from the graphics terminal

Parameters and variables:

ID,MABEL,LABEL,IB,MT - as previously described

LABL - return address after a scan operation

IJ - operation selection. 1 - SETPOT; 2 - SETLINES; 3 - TEST
4 - POTSET; 5 - COMPUTE; 6 - HOLD; 7 - RESET; 8 - SCAN
Descriptions of these operations are given in Appendix A


```

SUBROUTINE GRAPHAL(MABEL,LABEL,IB,NT,LABEL)
COMMON IU,II,INDEX,IPLST,ILP,ITR,IC,C9F,SC9F,RC9F,SNR,SP9W,NP9W,Q
REAL NP9W
NT=EXTEND IB(IU),NT(IU),LF(3)
DO 231 T= (201,222,222,234,...,209)IU
203 DO 231 I=31,42
231 IB(I)=0
DO 232 I=2,24,2
RECODE(2,227,IE(I-1))L
IF L.EI.O,GO TO 232
IF L.LF.1.O,L.GT.O32,RETURN MABEL
IF IC.O.O,IE(30+I/2)=TEST(L);GO TO 232
RECODE(2,227,IE(I))LL
CALL SLLIES(LL)
CONTINUE
IF IU.EG.2,RETURN LABEL
DO 233 I=2,24,2
IF I.(30+I/2).EQ.O,ENCODE(3,299,IB(I-1));GO TO 233
RECODE(3,209,IE(I))IB(30+I/2)
CONTINUE
CALL ITRIE
CALL ITR(10,IB,24,40,1,1,3,IE)
IF IE.L.O,OUTPUT(101)IE,'GRAPHAL-233-9'
IE(25)=77777777E
CALL ITR(10,IB(25),1,32,5,1,1,IE)
IF IE.L.O,OUTPUT(101)IE,'GRAPHAL-233-9'
IF IE.GT(1),I).EQ.O,GO TO 234
RETURN LABEL
CALL PILEI;RETURN LABEL
204 CALL SLLIFC(3,1,9,1);CALL COMPUTE;RETURN LABEL
205 CALL SLLIFC(3,-1,9,1);CALL HOLD;RETURN LABEL
206 CALL PILEI
207 CALL PILEI
208 ADD(3) 201 TO LF(1)
ADD(3) 202 TO LF(2)
ADD(3) 203 TO LF(3)
DO 221 I=1,23,2

```



```

221 ENCODE(5,LF(IJ),IB(I))
CALL TEXT3(10,18,24,39,1,1,3,IE)
IF IE.EQ.0,OUTPUT(101)'GRAPHAL-221'
22 222 I=1,24
222 I(I)=7777777
CALL TEXT3(10,18,1,40,1,1,3,IE)
IF IE.EQ.0,OUTPUT(101)'GRAPHAL-222'
223 IF (I.EQ.0)6.EQ.0,64 TO 223
CALL TEXT3(10,18,24,3,40,IE)
IF IE.EQ.0,OUTPUT(101)'GRAPHAL-223'
22 224 I=1,24,203,203)IJ
LF=40000;LI=4000 ;LX=40000
22 225 I=1,23,2
PFCODE(4,224,IP(I))L;DECODE(4,295,IB(I))LL
IF LL.EQ.0,IP(I).LL.NE.LI,RETURN MABEL
LL=LX(LLS(IB(I),6),6);PFCODE(4,296,LL)LL
IF LL.EQ.0,AND.L.NE.LX,6 TO 225
IF LL.LI.0,IP.LL.GT.057.0R.MBD(LL,10).GT.7,RETURN MABEL
DECODE(4,224,IP(I+1))J
I=LS(LA(J,77000000),12)
I2=LS(LA(J,77000000),12)
I3=LS(LA(J,77000000),6)
I4=LA(J,773)
REPEAT 124, FOR IR=11,12,13,14
IF IR.LI.0,IP.IR.GT.11,RETURN MABEL
1224 CONTINUE
PFCODE(4,226,IP(I+1))J
IF J.LT.0,IR.J.GT.9999,RETURN MABEL
CALL SETPAT(L,J)
225 CONTINUE
227 CALL SETLIES(5,1,9,-1);CALL RESET(100);RETURN LABEL
228 CALL ANRCAN(LABEL,18,15(1225),18(1425),18(1700),18(1750))
229 RETURN MABEL
230 IF (I.EQ.0)221 VAL ' )
231 IF (I.EQ.0)222 VAL ' )
232 IF (I.EQ.0)223 VAL ' )

```



```

293 FORMAT('IST VAL ')
294 FORMAT(44)
295 FORMAT(41)
296 FORMAT(14)
297 FORMAT(12)
298 FORMAT(' ')

```

```

END

```


SUBROUTINE ANOSCAN

Purpose: scans the addressable analog components in the actual computer mode and displays the result on the screen; after this, if selected, samples only the potentiometers, with the analog computer in the potset mode.

Parameters:

JK - return address

ITEX - text buffer

INAME - component names table

IVALUE - component readings storage

ID, MG, MT - as previously defined


```

ITEMP=480000;I=87
IF(LEAT 908, 500 L=0,64,128,192,256,320,16384,16448,16512
L=L+7
DO 907 J=L,1
  INCODE(4,R3,ITEMP)J
  IVALUE(I)=LIEP(ITEMP,ITEMP)
907 I=I+1
908 CONTINUE
909 K=1;K2=826
910 IF K1 E.1,K=1
DO 909 I=4,1005
  ITX(I)=777777779
DO 910 I=2,15675
  IF I.EQ.156,I=154
910 CALL SCAN(INAME(I),IVALUE(I),INAME(I+1),IVALUE(I+1),INAME(I+2),
CIVALUE(I+2),INAME(I+3),IVALUE(I+3),INAME(I+4),IVALUE(I+4))
DO 911 I=2,5-4,825,5
  U=(I+4)/5
  INCODE(4,51,ITX(I+3))IVALUE(U)
  ITX(I)=ITX(I+2)=ITX(I+4)=4H
  ITX(I+1)=IVALUE(U)
911 IF U.GI.1E3,ITX(I+1)=ITX(I+3)=4H
  INCODE(56,52,ITX(976))
  CALL DXTIT(ID,NG,2,IE)
  IF IE.GI.0,UTPUT(101)IF,'SCAN-DGINIT'
  CALL DTINIT(ID,MT,42,IE)
  IF IF.GI.0,UTPUT(101)IE,'SCAN-DTINIT'
DO 912 I=1,576,25
  CALL TEXP(ID,ITX(I),24,(I+24)/25,1,1,3,IF)
  IF IE.GI.0,UTPUT(101)IE,'SCAN-912 -I',I
912 CONTINUE
  CALL TEXP(ID,ITX(1001),1,40,80,1,1,IE)
  IF IE.GI.0,UTPUT(101)IE,'SCAN-AFTER 912'
913 IF(LEAT(1(41),9).EQ.0)99 19 913
  CALL TEXP(ID,ITX(1002),1,40,80,IE)

```



```

IF IE=0,OUTPUT(101)IE,'SCAN-913'
CECDE(4,53,ITEX(1002))ITEX(1002)
ITEX(1003)=4UP91S;ITEX(1004)=4HREIN
IF ITEX(1002).EQ.ITEX(1003),CALL P9TSET;K=87;IJ=1;GO TO 940
IF ITEX(1003).EQ.ITEX(1004),IBIT=1;IJ=4;RETURN JK
GO TO 930

```

F

SUBROUTINE ANALYSIS

Purpose: displays questions and decodes the answers to specify further the selection made; when a sample operation is made or after the other operations (except POWER and DISPLAY) stores the results in backstorage; rearranges the functions before calling FASTBRG.

Parameters and variables:

ID, MABEL, LABEL, M, YS, YR, YC, VAGO, IB, MT - as previously defined

IENT - pointer to indicate if the subroutine is entered after an operation is selected or reentered after FASTCOR.

IJ - pointer to the selection. 1 - SAMP; 2 - FFTR; 3 - CORR; 4 - CONV;
5 - POWER; 6 - DISPLAY; 7 - AUTO.
Descriptions of these operations are given in Appendix A.

II - used to save status when FASTCOR is called through FASTBRG.

INDEX - type of sampling operation

ICOUNT - counter to be used in FASTCOR; set to zero,

LBG - storage for the number of points displayed.

IMEM - storage for the present size of the correlation function kept in backstorage.


```

51090 JLINE ANALYSIS(MABEL,LABEL,YS,YR,YC,VAG9,IB,MT,M,IENT)
CMM9A IJ,II,INDEX,IPLT,I LP,IIR,IC,C0F,SC0F,RCEF,SNR,SP9W,NP9W,Q
CMM9B LOG(3),LIFE,YAC,ARRAY,RTH,IC9UNT,IMEN,PR9B
CMM9C IGA9B,IDI9T,ISTA,ISTB,ISTC,ISTD,ISTE
51100 NPP9
51110 ANALYSIS Y9(1),YR(1),YC(1),VAG9(1),IB(1),MT(1)
IC9=INT=0
IF I=1,LP=4,MY9=MY2+MY;LP=24 ;IV=ISP=1
IF IEV1,79,0,79 TO 300
IF IU,23,5,1=II;U=3;SC TO 1325
51120 MY9=MY+IE9(3);MY=X3
51130 IF I=2,3
51140 IF II,10,10,MY9=MY2+MY3;MY=0
51150 IF 201 I=1,MY
51160 Y99(I)=Y9(MY+I)
51170 Y99(3)(Y9(I),I=1,3*MY),(YC(I),I=1,2*MY),(YR(I),I=1,M)
51180 IF 202 I=1,MY
51190 Y9(MY+I)=Y99(I)
51200 IF 1200 I=MY+1,MY
51210 Y9(I)=0.
51220 IF I=1,3
51230 Y99(3)(Y9(I),I=1,3*MY),(YC(I),I=1,2*MY),(YR(I),I=1,M)
51240 RETURN LA3TL
51250 IF IE(101,302,303,302,302,302,302,309,309)IJ
51260 Y99DE(40,301,13)
51270 CALL TEXT9(IC,12,10,39,1,1,3,IE)
51280 IF IE,12,0,0,IPUT(101)IE,'ANALYSIS-301-9'
51290 Y9(1)=7777777;CALL TEXT9(IC,13,1,40,1,1,3,IE)
51300 IF IE,12,0,0,IPUT(101)IE,'ANALYSIS-301-R'
51310 IF 990(10,0,2),83,0,99 TO 311
51320 CALL TEXT9(IC,13,1,40,40,IE)
51330 IF IE,12,0,0,IPUT(101)IE,'ANALYSIS-311'
51340 Y99DE(2,302,13)13
51350 IF IE(1),1,LP,RETURN
51360 IF I=3A,1,3,00 TO 2311

```



```

1311 I=1,M
1311 YS(I)=YS(I+M2)
1311 CALL XTRCA 'P'(M,YS,YR)
1311 GO TO 305
1311 EXECDE(32,793,IB)
1311 CALL TXTRP(13,13,13,33,1,1,3,IE)
1311 IF IE.C.0,OUTPUT(101)IF,'ANALYSIS-321'
1311 GO 322 I=1,24
1311 IS(I)=777777777
1311 CALL TXTRP(10,10,24,40,1,1,3,IE)
1311 IF IE.C.0,OUTPUT(101)IF,'ANALYSIS-322'
1311 IF IE.C.0,ES,0,30 TO 323
1311 CALL TXTRP(13,13,13,33,1,1,3,IE)
1311 IF IE.C.0,OUTPUT(101)IF,'ANALYSIS-323'
1311 GO 324 I=1,5,2
1311 EXECDE(1,324,13(I))J
1311 IF J.E.1,J=(I+1)/2;GO TO(309,325,1333,304,305,306,307)IJ
1311 GO TO 324
1311 IF J.E.3,7=I*5/2
1311 CALL EXECDE(13,M1,LABEL,YS,YR,VAG3(M+1),IB,M)
1311 EXECDE(32,306,IE)
1311 GO TO 321
1311 IF J.E.3,I=10
1311 CALL EXECDE(13,M1,LABEL,YS,YR,VAG3,IB,M,J,M1,M2,ISP,1)
1311 IF IV.C.1,IV=2;J=J;GO TO 303
1311 IF J.E.3,I=10;GO TO 333
1311 GO TO(341,333,343)JJ
1311 GO 342 I=1,1
1311 YS(I)=YR(I)
1311 GO TO 323
1311 GO 344 I=1,10
1311 YS(I)=YC(I)
1311 GO TO 323
1311 GO TO 3

```



```

IF IV.EG.1,CG TO 304
IF J.E.3,RETURN LABEL
IF JJ.EG.3,M2=IMEM
IF J.E.1,AN.0,JJ,NE.1,M1=N2
II=1/2,IF J.E.2,II=M
IF J.E.2,R,0,0,EG.1,M2=M1
IF (2+1,251,353)JJ
1305 CG TO 302 I=1,M2
351 CG TO 352 I=1,M2
352 YR(I)=Y(I)
CG TO 353
353 CG TO 354 I=1,M2
354 YR(I)=YC(I),CG TO 343
355 CALL DISPLAY(YR,Y5,YC,VAS0,M,J+3,18,0)
356 RETURN LABEL
357 IF IV.EG.1,CG TO 304
IF J.E.3,RETURN LABEL
IF J.EG.0,AN.0,0,0,EG.1,M2=M1
CG TO 1305
1305 RETURN LABEL
1306 FORMAT('IF CORRECT CONST RETN: ELSE TYPE - N0- ')
1307 FORMAT(A2) SIGN C9R ENTER 1 UNDER FIRST LETTER')
1308 FORMAT('REPL PERIOD RECUR ENTER 1 UNDER FIRST LETTER')
1309 FORMAT(M1)
1310 FORMAT('IF 100

```

END


```

SUBROUTINE NORMALIZE(YA,X,K)
COMMON IJ,II,INDEX,IPLT,ILP,ITR,ID,COF,SCOF,RCOF,SNR,SPOW,RPOW,Q
REAL NPOW
DIMENSION YA(K)
TEMP=0.
DO 801 I=1,K
801 TEMP=AMAX(ABS(YA(I)),TEMP)
COF=X/TEMP
DO 802 I=1,K
802 YA(I)=YA(I)*COF
*
RETURN;END

```

Purpose: normalizes the maximum absolute value of a function to a preset limit.

Parameters:

YA - initial storage address of the function

X - maximum value

K - number of points

COF - normalization factor

SUBROUTINE SEGSAMP (and XEGSAMP)

Purpose: controls the sampling operation and performs the normalization required for each type of processor.

Parameters and variables:

M,YS,YR - as previously defined.

MAC - counter for the samples

ARRAY - location of the sample taken by SAMPLE

ISAMP - type of processor. 0 - linear processor; 4 - clipper-limited processor;
8 - amplitude limited processor.

INDEX - type of sampling operation. 1 - samples M times and stores as REPLICA (YR);
2 - samples 3M times and stores as SIGNAL (YS); 3- samples 2M times and
stores as the last samples of SIGNAL

COF - normalization factor last computed

RCOF - normalization factor for REPLICA

SCOF - normalization factor for SIGNAL

Q - ratio of RCOF to SCOF


```

SUBROUTINE SEGSAMP(M,YS,YR)
COMMON IJ,IJ1,INDEX,IP,IP1,ILP,ITR,ID,C0F,SC0F,RC0F,SNR,SP0W,NP0W,C
COMMON LFG(3),LINE,MAC,ARRAY,RTH,ICOUNT,IMEM,PR0B
COMMON ISA,P,IDIST,ISTA,ISTB,ISTC,ISTD,ISTE
REAL NP0W
DIMENSION YS(1),YR(1)
JA=M+1;JC=JA+1
IF INDEX.EQ.3,CALL SETLINES(8,1,9,-1)
CALL RIGHT(100)
IF INDEX.NE.1,GO TO 203
201 MC=MAC=0
CALL SETLINES(30,-1,8,1,9,1)
202 IF MC.EQ.MAC,GO TO 202
YR(1-MAC)=ARRAY
MC=MAC
IF MAC.GT.C0B TO 202
CALL SETLINES(8,-1,9,1,20,1)
CALL NORMALIZE(YR,.3,M)
C0F=C0F
IF ISA.P.EQ.4,GO TO 206
203 MAC=JC;IF INDEX.EQ.3,MAC=JA
MC=MAC
CALL SETLINES(30,-1,8,1,9,1)
204 IF MC.EQ.MAC,GO TO 204
YS(JC-MAC)=ARRAY
MC=MAC
IF AC.CE.C0B TO 204
CALL SETLINES(3,-1,30,1)
IF ISA.P.EQ.4,C=1;GO TO 208
IF ISA.P.EQ.8,C=1;GO TO 210
IF INDEX.EQ.2,CALL NORMALIZE(YS,.3,JC+1);SC0F=C0F;0=RC0F/SC0F;
C0F=C0F
205 206 I=I+1;JC+1
205 YS(I)=VS(I)*SC0F

```



```

RETURN
206 207 I=1,JA+1
    IF YR(I).GT.C,YR(I)=.3
207 IF YR(I).LE.C,YR(I)=C.
    RETURN
208 209 I=1+(INDEX/3),JC+1
    IF YS(I).GT.C,YS(I)=.3
209 IF YS(I).LE.C,YS(I)=C.
    RETURN
210 IF INDEX.EQ.3,209 TO 212
211 YS(I)=YS(I)/SCOF
212 CALL MEXCAL IZF(YS,.3,JC+1);SCOF=COF
    RETURN;END

```


SUBROUTINE DISPLAY (and TISPLAY)

Purpose: prepares the functions and sends them to display on the graphics terminal

Parameters and Variables:

YA,YB,YC - storage for the functions to be displayed

VAGO,IB - buffers

M - sample block size

MI - pointer to indicate how many points of each function are to be displayed and the display option. MI=1 indicates that YA and YB have M points and YC three times M points and the y-axis is to be at the left side of the screen; MI=2 is the same as above but with YB having 2M points; MI=3 still the same but with 2M points for all three functions; MI=4, 5, or 6 means that YA, YB or YC, respectively is to be displayed isolated with the maximum value normalized to .7; MI=7 indicates that all three functions have 2M points and the y-axis is to be placed at the center of the screen.

IREC - pointer to indicate that with the options 1, 2 or 3: only the first function and the three sets of coordinate axes are to be shown (IREC=1); only the last two functions are to be displayed (IREC=2); normal option (IREC=0).

LBG(1,2,3) - stores the number of points displayed for YA, YB and YC

ID - terminal device number


```

SUBROUTINE FDISPLAY(YA,YB,YC,VAGS,M,M1,I8L8CK,I8EC)
COMMON IJ,I1,INDEX,IPLST,IPL,IIR,ID,C0F,SCGF,RC9F,SNR,SP0W,NP0W,Q
COMMON I8G(3),LINE,MAG,ARRAY,RTH
CVAL=V0.
IF(I8L8CK) VAGS(M),VC(M),VAGS(M),I8L8CK(M)
JA=I+M1;IN=I;X=XX=-.0;Y=0.;IT=7;YU=1.;YL=.4
I8L8CK(1)=I8EC(0,10)
DO I8(101,100,103,110,111,112,115)M1
M1=VC=M1;M3=JA;G6 T0 104
M1=M2=M3=JA;G6 T0 104
M1=VC=M3=JA
IF I1 T5:12,IN=2;M1=M1/2
IF I2:GT.5;P.83.M3.GT.512,IN=2;M3=M3/2
CY=1.0/12
IF I3:GT.7;GX=2.0/3;X=-1.
IF I8EC.E0.1,G6 T0 2100
DO 105 I=1,M1
IL=I;IF I.G.1,IL=0
I8L8CK(I+1)=I8L8CK(X+DX*I,YA(I*M)+.7,IL)
CALL IYVE
I8L8CK(1)=1
CALL GMAPM(ID,I8L8CK,M1+1,1,IE)
IF(IE.E.0)BUTPUT(101)IE,'DISPLAY-105'
IF I8EC.E0.2,G6 T0 116
DO 106 I=1,M2
IL=I;IF I.E.1,IL=0
I8L8CK(I+1)=I8L8CK(X+DX*I,YB(I)+.05,IL)
CALL IYVE
L1(2)=5
CALL GMAPM(ID,I8L8CK,M2+1,2,IE)
IF(IE.E.0)BUTPUT(101)IE,'DISPLAY-106'
DO 107 I=1,M3
IL=I;IF I.E.1,IL=0
I8L8CK(I+1)=I8L8CK(X+DX*I,YC(I*IN)-.6,IL)
DO 107 I=1,M3

```



```

110(3)=N3
CALL GRAPHT(ID,IBLCK,N3+1,3,IE)
IF(I*.01)OUTPUT(101)IE,'DISPLAY-107'
IF YA(JA-1)*.E..3333,GO TO 116
IFLCK(1)=IHEAD(1,10)
IFLCK(2)=IPACK(-.5,YA(JA)-.6,0)
IFLCK(3)=IPACK(1.,YA(JA)-.6,1)
CALL GRAPHT(ID,IBLCK,3,4,IE)
IF IE*.01)OUTPUT(101)IE,'DISPLAY-107-1HR'
YA(JA)=YA(JA-1)+.0.
115 IF IDEG*.01,STOP
IFLCK(1)=IPACK(0,2)
I=3; REPEAT 117, WHILE I.LE.IT
IFLCK(2)=IPACK(-1.,.7-Y,0);IBLCK(3)=IPACK(1.,.7-Y,1)
IFLCK(4)=IPACK(XX,YC-Y,C);IBLCK(5)=IPACK(XX,YL-Y,1)
CALL GRAPHT(ID,IBLCK,5,1,IE)
IF IE*.01)OUTPUT(101)'DISPLAY-116'
Y=Y+.6;
117 I=I+1
GOTO 115
118 I=110; I=1,M
119 Y=0(I)=YA(I);Y=N;GO TO 113
120 I=111; I=JA
121 Y=0(I)=Y=JA;GO TO 113
122 I=112; I=JA
123 Y=0(I)=YC(I);N=JA
124 CALL GRAPHT(I7F(VAGG,.7,N)
X=1.0X
125 I=14; I=1,F
126 IL=1;IF I.C.1,IL=0
127 I=1;IF I=1,IBLCK(I+1)=IPACK(X+DX*I,VAGG(I)+.05,IL)
CALL GRAPHT(ID,IBLCK,1+1,2,IE)
IF I*.01)OUTPUT(101)IE,'DISPLAY-114'
128 I=1;IF I.C.3)=0;LEG(2)=N
Y=.5;I7=5;YJ=1.55;YL=-.25;GO TO 116

```


115 8X=0.12 10 103

200

*

SUBROUTINE SEGFOUR

Purpose: prepares the function to be transformed by the library program FOUR2 and rearranges the result for displaying

Parameters and variables:

MF - pointer to indicate which function is to be transformed

M - sample block size

LABEL - return address

YA,YB,YC - function locations; YA is assumed to be of size M and the other two 2M

FT - complex buffer for the transform operation

IB - buffer

K - indicates, if 0, that display is not required


```

SUBROUTINE SECF9UR(MF,N,LABEL,YA,YB,YC,FT,IB,K)
COMMON JU,II,INDEX,IPLST,IPL,I TR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,0
REAL N9C
NY ENSURE YA(1),YB(1),YC(1),IB(1)
COMPLEX FT(1)
N=JA=4
DO TO (901,902,903)MF
901 DO 951 I=1,N
951 YA(I)=YC(I);N=N;GO TO 902
902 DO 953 I=1,JA
953 YA(I)=N*(I)
903 DO 901 I=1,N
901 FT(I)=CMPLX(YA(I),0.)
CALL FCF2(FT,N,1,-1)
L=N/2;DO 902 I=1,L
902 FT(I)=CABS(FT(J))
YC(I)=ATANG(ATMAG(FT(J)),REAL(FT(J)))
YC(J)=ATANG(ATMAG(FT(I)),REAL(FT(I)))
CALL NORMALIZE(YC,.3,N)
CALL NORMALIZE(YC,.3,N)
CALL TISPLAY(Y,YB,YC,YC,N/2,7,IR,0)
951 N=3
READ (5)(YA(I),I=1,3*N),(YB(I),I=1,2*N),(YC(I),I=1,M)
RETURN LABEL:END

```


SUBROUTINE FASTCOR

Purpose: performs a very fast correlation or convolution evaluation, considering both functions periodic or both aperiodic; may also consider one of the functions as one of very long duration and then the aperiodic correlation is computed by the partition method.

Parameters and variables:

IT - pointer to indicate the option, according to: IT=1 - periodic; IT=2 - aperiodic; IT=3 - recursive, where the partition method previously described is employed recursively with two new samples being read in every recursion.

YA - address of array where the second function (signal) is stored

YB - address of array where the first function (replica) is stored and where the results are placed

YC,YD - address of arrays of reals used for temporary storage

FT,FF,FW - address of complex arrays used to prepare the functions for transformation and receive the results from FOUR2

IB - buffer to be passed to DISPLAY

MR - size or number of samples of the first function (replica); has to be a power of two

MS - size of the second function (signal); has to be a power of two and greater than or equal to MR (the function actually has another M block in excess to MS if the partition method is used).

ISP - pointer that if 0 indicates that display is not required

MT - address of text directory

LABEL - return address for the subroutine

M - basic sample block size

IJ - if 4 indicates a convolution; if 3, 5 or 7 asks for correlation

ITR - normalized threshold adjusted by the operator

ID - graphics terminal number to be used when calling text display

SNR - input signal-to-noise ratio

Q - ratio of the normalization factor for replica to that of signal

RTH - display normalized threshold

ICOUNT - counter for the number of recursions; if an automatic stop for the recursion is desired after n sections, ICOUNT has to be preset to 10000-n

PROB - probability of detection when computed by the subroutine, or false alarm rate

IDIST - individual sample number when threshold was last reached

- ISTA - signal injection potentiometer adjustment, used to compensate the normalization factor in a recursive option
- ISTC - indicator to variations in the recursive option. If equal to 3 means that probability of detection is to be computed and an approach equivalent to the partition method permits getting the aperiodic cross-correlation of two different samples in one computation. If equal to 4 means that distance measurement is to be made and then the actual sample number is displayed whenever the threshold is reached. If equal to 5 indicates that false alarm rate is to be evaluated.
- ISTD - counter for the text block number where the distance is to be displayed
- ISTF - indicator to the value of ICOUNT at which the signal is to be added to noise when in distance measurement


```

5006 CONTINUE PASTCER(IT,VA,YA,YC,YD,FT,FF,IB,MR,YS,ISP,MT,LABEL,M)
5007 CALL IJ,II,INDEX,IPLT,IPL,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,G
5008 CALL Y2V LHC(3),LINE,YAC,ARRAY,RTM,ICOUNT,IMEN,PR93
5009 CALL YV ISAMP,DISI,IST,I,ISTB,ISTC,ISTD,ISTE
5010 CALL YSN,ISTF,IK1,SPR(16C)
5011 CALL VFOR
5012 IF (NVSIN VA(1),Y2(1),YC(1),YD(1),IB(1),MT(1)
5013 CRP-EX FF(1),FT(1),FX(1)
5014 COMPLEY SAVEFF
5015 J2=2
5016 IF Y=II=ICFT=0
5017 PA=PA+PAR=0.
5018 IF ISTC=EQ.4,II,IST=1
5019 IF IT=EQ.1,K=KR=KS=MF
5020 IF IT=EQ.2,ANU,MP,LT,MS,KR=NR/2,KS=YS-KR,K=MS
5021 IF IT=EQ.2,AND,NR,EQ,MS,KS=KR=NR,K=KR+KS
5022 IF IT=EQ.2,KR=NR,K=KS=YS,K=KS-KR,K=3;IND=INDEX;INDEX=3;
5023 CALL SETLIPS(14,-1)
5024 IF ISTC=EQ.2,K=K;IK=1
5025 XI=X+1;FIC=1./X;L=X/2
5026 C=2*K+2;K=K,-1
5027 DO 501 I=K+1,K
5028 YC(I)=0.
5029 DO 503 I=1,K
5030 Y2(I)=YC(I-1)
5031 IF IJ=EQ.4,99, I+ 505
5032 DO 504 I=1,K
5033 FF(I)=CALV(Y2(I),C.)
5034 IF ISP=EQ.2,99, I+ 507
5035 CALL DISPLAY(YC,YB,Y2,Y4,L,3,IB,2);99, I+ 507
5036 DO 506 I=1,K
5037 FF(I)=COMPLX(YC(I),0.)
5038 IF ISP=EQ.2,99, I+ 507
5039 CALL DISPLAY(Y,YC,YC,YC,L,3,IB,2)
5040 CALL COMPLEY(FF,K,I,-1)

```



```

IF IT*E*3,60 TB 603
DO 608 I=1,K
508 FT(I)=C*PLX(YC(I),O.)
IV=4750 I= 606
609 CALL NORMALIZE(YB,3,K)
YC=F*COF
IF ICAMP.E).0,CCF=COF*9000.*B/ISTA
CY=TH=IT*3.E-4
IF (STC.EC*3,60 TB 1601
601 DO 602 I=1,K
602 FT(I)=C*PLX(YA(I),YA(K+I))
603 I=K+1
1601 DO 1602 I=1,KR
1602 FT(I)=C*PLX(YA(I),YA(I+KR))
603 I=KR+1,K
1603 FT(I)=C*PLX(C,O.)
604 I=K
604 DO 604 I=1,KR
604 FT(I)=C*PLX(YA(I),O.)
605 I=K+1,K
605 FT(I)=C*PLX(C,O.)
606 CALL FT*CR(FT,K,1,-1)
607 I=1,K
607 FT=REAL(FT(I))AI=AIMAG(FT(I))
608 FT=REAL(FT(I))AJ=AIMAG(FT(I))
609 FT=RE*FA-AI*AJAI=RE*AJ+AI*RE=RD
607 FT(I)=C*EL*(RE,AI)
CALL F*CR2(FT,K,1,I)
608 I=1,L;J=I+L
YC(I)=(REAL(FT(I)))*FIC
YC(J)=(AI*AC(FT(I)))*FIC
609 I=1,L;J=I+L
YB(J)=YC(I)
YC(I)=(REAL(FT(J)))*FIC
609 YC(I)=(AI*AC(FT(J)))*FIC

```



```

IF IV=EQ.1, IV=Q:G9 T9 509
IF IU=K.4, G9 T9 611
IF IF=Q.1, G9 T9 611
C9 610 I=1, L:J=L+I
TE P=YI(I):SEMP=YC(I)
YI(J)=YI(J):YC(I)=YD(J)
YI(J)=TE.P
YI(J)=SEMP
611 IF II=Q.2, G9 T9 701
CALL V=VAL IZE(Y=,3,K)
IF IS=Q.1, RETURN LABEL
I=5:IF K:ST.MS,I=1
CALL DISPLAY(YA,YA,YB,YE,L,I,IS,1)
RETURN LABEL
701 IF ISIC=Q.3, G9 T9 1701
IF ISIC=Q.4, G9 T9 2701
E4 1702 I=1,KK
YD(I)=YI(K+I)*XC9F
1702 IF YE(I).Q.THR, IDET=1:FAR=FAR+1.
C9 702 I=1,KK
YI(K+I)=YI(K+I)*XC9F
702 IF YI(K+I).SE.THR, IDET=1:FAR=FAR+1.
IF ISIC=Q.5, PR9H=FAR/(QY*KK)
C9 T9 703
2701 IS(2)=4F
IL 99=1
C9 2702 I=1,KK
YI(I)=YI(K+I)*XC9F
IF YI(I).SE.THR, IDET=1:G9 T9 2704
2702 IDIST=IDIST+1
IL 99=2
C9 2702 I=1,KK
YI(K+I)=YI(K+I)*XC9F
IF YI(K+I).SE.THR, IDET=1:G9 T9 2704
2703 IDIST=IDIST+1

```



```

IF ISTF-ICOUNT.EQ.1,CALL SETLINES(16,1)
IF ISTF.EQ.ICOUNT,CALL SETLINES(16,-1)
G0 FR 703

2704 ENCODE(4,927,13)IDIST
CALL TEXTU(1,18,2,ISTD,1,1,3,IE)
IF IE.M.O.OUTPUT(101)IE,'FASTC9R-2704',
ISTD=ISTD+1
IF ISTF.GT.40,ISTD=1
IF IL999.EQ.1,40 FR 2702
G0 FR 2703

1701 FR 3702 I=1,KK
YB(I)=YB(I)*XCF
YB(I+K)=YB(I)*XCF
IF YB(I).GE.THR,IX=1
IF YB(I+K).GE.THR,IX=1
IF IX.EQ.1,IX=C;PR9BA=PR9BA+1;IDET=1
IF IX.EQ.1,IX=C;PR9BA=PR9BA+1;IDET=1
PR9B=PR9BA/KK
703 ICOUNT=ICOUNT+1
IF ISD.EQ.7,92 FR 704
IF IU.C.O.4,CALL DISPLAY(YA,YA,YB,YC,KK,3,14,1);G0 FR 706
SAVEFF=FF(KK)
FF(KK)=CMPLX(.3323,THR)
CALL DISPLAY(FF,YA,YB,YC,KK,IK,15,1)
FF(KK)=SAVEFF
D0 704 I=1,24
704 IS(I)=77777777F
CALL TEXTU(10,18,1,28,1,1,1,IE)
IF IE.M.O.OUTPUT(101)IE,'FASTC9R 704 AP'
IF IDET.EQ.CENCODE(24,991,IP)
IF IDET.EQ.1,ENCODE(44,992,18);ENCODE(8,993,18(12))PR9B
CENCODE(12,994,19(14));ENCODE(6,993,18(17))THR
CENCODE(8,996,18(12));ENCODE(8,993,18(21))SNR
CALL TEXTU(10,18,23,28,1,1,3,IE)
IF IE.M.O.OUTPUT(101)IE,'FASTC9R-704',

```



```

706 IND=0
IF TEST(14).EQ.1.OR.ICOUNT.EQ.10000,INDEX=IND;GO TO 801
05 705 I=1,KS/2
708 YA(I)=YA(KS+I)
CALL SESSAMP(MR,YA,YC)
JK=JK+2
IF JSTC.EQ.3,GO TO 1601
04 706 801
CALL SETLINES(14,1,16,1)
IF ISQ.EQ.CCR.IJ.EQ.4,RETURN LABEL
WRITE(6,995)PROB,THR,JK,SNR
CALL TEXTR(10,18,23,38,5,1,3,IE)
CALL SETLINES(14,1)
IF IE.F.C.OUTPUT(101)IE,'FASTCR-801'
802 IF ASD(F(1),3).EQ.C,GO TO 802
04 803 I=1,24
803 IS(I)=77777777
CALL TEXTR(10,18,23,38,5,1,3,IE)
IF IE.F.C.OUTPUT(101)IE,'FASTCR-803'
RETURN LABEL
991 FORMAT('THRESHOLD NOT REACHED. ')
992 FORMAT('THRESHOLD REACHED.PROBABILITY OF DETECTION: ')
993 FORMAT(F8.4)
994 FORMAT(' THRESHOLD:')
995 FORMAT('O PROBABILITY OF DETECTION ',F8.2,6X,'THRESHOLD',F8.4,
6X,'SIGNAL LENGTH',18,' SECTIONS',6X,'SNR',F8.4)
996 FORMAT(' SNR ')
997 FORMAT(14)
END

```


SUBROUTINE PROCESS

Purpose: establishes a sequence of operations to compute input signal-to-noise ratio, input-output performance, receiver operating characteristics, or distance measurement.

Parameters and variables:

Arguments - as in ANALYSIS

IS - after used as a pointer to the selection, is used to save status

II - pointer to indicate the position where the loop is to be resumed when the subroutine is reentered

INDEX - pointer to indicate type of sample

ITR - normalized threshold

COF - last normalization factor computed

RCOF - last REPLICA normalization factor computed

SCOF - last SIGNAL normalization factor computed

SNR - signal-to-noise ratio

SPOW - signal power

NPOW - noise power

Q - ratio of REPLICA normalization factor to SIGNAL normalization factor



LBG - used to save status
 ICOUNT - counter to the number of recursions to be used in FASTCOR
 IMEM - location where the number of points computed in performance determination or the number of SNR values used in ROC determination is stored
 PROB - used to bring from FASTCOR the probability of detection or false alarm rate; also used to save status
 ISAMP - as in SEGSAMP
 IDIST - used to save status or, in ROC determination, to store the number of threshold values used
 ISTA - signal injection potentiometer setting
 ISTB - noise injection potentiometer setting
 ISTC - stores the pointer to the selection, necessary for use in FASTCOR or PISPLAY
 ISTD - used to save status
 ISTE - step size for decreasing ISTA and increasing ISTB
 ISTF - in distance determination used to transfer the burst occurrence counter setting to FASTCOR, otherwise saves status


```

SUBROUTINE PRECESS(MABEL,LABEL,YS,YR,YC,VAG9,IB,MT,M,IENT)
COMMON IU,II,INDEX,IPLT,ILP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,O
COMMON LCO(3),LINE,MAC,ARRAY,RTH,IC8UNT,IMEN,PR9B
COMMON ISATP,ISIGT,ISTA,ISTB,ISC,ISTD,ISTE
COMMON ISIF,IKT,SPR(160)
      F.L=IP9W
      IF EXGIER YS(1),YR(1),YC(1),VAG9(1),IB(1),MT(1)
      IF IENT.EQ.O,GOTO 1000
      DO 10 I=1,304
        CALL SETUI-ES(25,-1,25,1)
        CATCH (100,200,300,400)IU
        INPUT SIGNAL-TO-NOISE RATIO (AFTER TRANSDJCR) *****
        100 ISIC=1
        CALL GETIT
        CALL SCAN(4HPC23,ISTA,4HF022,ISTB)
        1100 TOL=0
        1101 CALL SETPGI(4HP022,O,4HP023,ISTA)
        II=IU+1;INDEX=2
        IPT=IUA/P;ISAMP=0
        REPEAT 103,WHILE IU.LT.R
        CALL XPSGA P(Y/Z,YS,YR)
        PR=YC+X*Y
        DO 1101 J=1,P
          YS(J)=YB(I)
        CALL EASTERG(MABEL,LABFL,YS,YR,YC,VAG9,IB,MT,M,M,0,2)
        101 TEND=0.
        SC9F=SC9F+C9F
        DO 102 J=1,2*X
          TEMP=MAX(YC(I),TEMP)
          102 I=TEMP/NP9B
          IF IO.EQ.1,SPR=NPOW;CALL SETPGT(4HP023,O,4HP022,ISTB)
          IO=IO+1
          SPR=10*ALBGLC(SPOW/NP9W+1.E-20)
          103 IP=P+INTO

```



```

IF ISTC.EC.1.AND.ISTC.EC.2.GE.19 201
IF ISTC.EC.1.AND.ISTC.EC.3.GE.19 302
IF ISTC.EC.2.GE.19 2201
IF ISTC.EC.3.GE.19 3302
CALL SETPR(4*PO22,ISTA,4*PO22,ISTB)
CALL SPILL(YS(25,1,26,-1)
RETURN LABEL
* R/C OR PERFORMANCE *****
200 ISTC=2
I=YEY=1
I=1
CALL SETPR(4*PO22,0,4*PO23,9000)
CALL XCSAMP(N,YS,YS(2*N+1))
PR=CAF
ISTF=1
C=1E-2100
201 NLOCAT 208,FILE SNR.GR.2
IF ISTC.EC.2.GE.19 3102
2201 CD=(IKI)=S/N
IKI=IKI+1
CALL XCSAMP(M/2,YS,YR)
IF=2
CALL FASTPRG(MABEL,LABEL,YS,YS(M*2+1),YC,VAGE,IB,MT,M,2,M,N,0,2)
202 TEMP=RT*SCF*CAF
DO 203 I=1,N
YC(I+1)=YC(I+N)/TEMP
203 YS(I)=YS(I+N*N)=YC(I)/TEMP
IF=3
WRITE(3)(YC(I),I=N+1,2*N)
IF=3
CALL FASTPRG(MABEL,LABEL,YS,YS(M*5+1),YC,VAGE,IB,MT,M,2,M,N,0,2)
204 NLOC=0.
IF=3
DO 205 I=1,2*N
205 PR=MAX(YC(I),PRG)

```



```

2013 PRG=PRG/CRF
2014 READ(3)(YS(I),I=1,M)
2015 I=1,M
2016 YX(I)=YS(I)
2017 IF=4
2018 CALL FASTTRG(MABEL,LABEL,YS,YR,YC,VAG9,IB,MT,M,M,0,2)
2019 TEMP=0.
2020 I=1,M
2021 IF=AVAY(YC(I),TEMP)
2022 CRF(IKT)=TEMP/CRF+PRG3
2023 IKT=IKT+1
2024 CALL SFTPT(4,PRG23,ISTP)
2025 CALL XRCFA'P(N/2,YS,YR)
2026 I=5
2027 CALL FASTTRG(MABEL,LABEL,YS,YS(M*2+1),YC,VAG9,IB,MT,M,M,1,0,2)
2028 TEMP=0.
2029 I=1,2,M
2030 IF=AVAY(YC(I),TEMP)
2031 CRF(IKT)=TEMP/PRH/SCRF/CRF
2032 IKT=IKT+1
2033 ISTA=ISTA-ISTE;ISTP=10000-ISTA
2034 IF IAT.EQ.156,STOP=-1
2035 ICRF=2
2036 IKT=IKT+1
2037 K=1
2038 I=1,IEN
2039 YS(I)=YS(I+IEN)=SPR(K)
2040 YC(I)=YC(I+IEN)=10*ALB010(SPR(K+2))*2/SPR(K+1)+1,E-20)
2041 I=I+3
2042 IOST=IEN
2043 CALL DISPLAY(YC,YR,YC,VAG9,IB,LABEL,MT)
2044 * *****
2045 IOST=3
2046 IF IEN .EQ. 3; ILEN=1
2047 IKT=1

```



```

CALL SETPT(4*P022,0,4*P023,9000)
CALL XEGSAMP(M,YS,YR)
CPY(160)=RCOF
LPT(1)=ITR
WRITE(3)(Y2(I),I=1,M)
IDIST=ITR/100;IF ITR-IDIST*100.GT.0,IDIST=IDIST+1
ITR=1
C=15.6100
300 REPEAT 307,WHILE S.R.GE.0
IF IDIF.F0.2,00 19 3100
302 CPY(IKT)=S.R
IKT=IKT+1
304 REPEAT 306,WHILE ITR.GT.0
ITR=1
ISIC=5
306 REPEAT 305,WHILE ISTD.LT.5
STD=STD
COUNT=COUNT+1
308 (3)(Y(I),I=1,M)
CALL XEGSAMP(M,YS,YR)
IF ISAVG.EQ.0,C=SDR(160)/SCUF
IL=0
CALL FASTRC(MAREL,LABEL,YS,YR,YC,VAG9,IB,MT,M,3,M,2*M,0,2)
304 SPT(IKT)=P.F.Y
IKT=IKT+1
IF ISTD.EQ.1,CALL SETPT(4*P023,ISTA);ISIC=3
306 ISTD=ICPD+1
308 (IKT)=ITR/100.
IKT=IKT+1
309 ITR=ITR-100
ITR=ICPD(1)
ICPA=ISTA-ISTE;ISTS=10000-ISTA
IF IKF.GE.15-2*IDIST,S.R=-1
ITR=2
307 ITR=ITR+1

```



```

J=X+1
LL=IDIST*PEV
KK=2
DO 300 I=1,I*EM
  YC(I)=SPR(K)
  DO 300 L=1,IDIST
    JJ=LL+J
    YS(J)=YS(JJ)=SPR(KK)
    YC(J)=YC(JJ)=SPR(KK+1)
    VAS(J)=SPR(KK+2)
    KK=KK+3
  300 JJ=J+1
  KK=KK+1
  300 K=I+3*IDIST+1
  ITEMP=TEMP; I*EM=IDIST; IDIST=ITEMP
  CALL PLOTFLY(YS,YR,YC,VAG9,I2,LABEL,MT)
  * DIGTA CE DETECTION *****
  400 ICGUNT=999
  YCIT=999
  IOTD=1
  ISTC=4
  IJ=3
  CALL SETFHT(4*PO22,0,4*PO23,9000)
  I*EX=1
  CALL XPGSAMP(X,YS,YR)
  CALL GETFHT(4*PO22,IST2,4*PO23,ISTA)
  I*EX=2
  CALL SETLI (S(16,-1))
  CALL XPGSAMP(X,YS,YR)
  CALL FASTG-Q(MADEL,LABEL,YS,YR,YC,VAG9,I3,MT,M,3,M,2*M,1,3)
  END

```


SUBROUTINE PISPLAY

Purpose: displays the results of performance determination or ROC evaluation

Parameters and variables:

YS - storage for input SNR or false alarm rate

YC - storage for output SNR or probability of detection

YR - storage for signal-to-noise ratio in ROC determination

VAGO - storage for threshold settings

IB,LABEL,MT - as previously described

LBG - storage for the number of points displayed

ISTC - indicator of performance determination (2) or ROC determination (3)


```

SUBROUTINE PISPLAY(Y5,Y6,YC,VAG9,IB,LABEL,MT)
COMMON IJ,II,IJDFX,IPL,OT,ILP,ITR,ID,CE5,SC9F,RC9F,SNR,SP9W,NP9W,Q
COMMON LQG(3),LINE,VAC,ARRAY,RTH,IC9JNT,IYEV,PR9E
COMMON IBA,P,IDIST,ISTA,ISTB,ISTC,ISTD,ISTE
COMMON ISTF,IKT,SPR(16)
CPL NQCA
CALL SILL'LS(25,1,26,-1)
TIMESTP=YS(1),YR(1),YC(1),VAG9(1),IB(1),MT(1)
IC=IYEV/IF ICIC=EQ,3,IK=IYEV*IDIST
CALL NVALIZE(YS,1,IK)
CALL NVALIZE(YC,3,IK)
IP(1)=IHFAP(0,10)
IP(1)=LQ3(3)=0:LQ6(2)=IK
DO 101 J=1,IK
  YE1
  IF YE2(IP,IPV),EQ,1,IK=0
  IF ICIC=EQ,3,YC(1)=-YS(1)
101 IP(I+1)=IPIC(Y5(I),YC(I),I)
  CALL QAPPN(IC,IB,IK+1,2,IE)
DO 103 I=1,24
  IF I(1)=45
  C9 IC2 I=1,IK
  YY=YC(I+IK)
  IF ISTC=EQ,2,FNCODE(4,990,IB)YY
  IF ISTC=EQ,3,FNCODE(4,995,IB)YY
  Z=I,T((9-YC(I))*30/1.8+.6)+5
100 CALL TEXT9(ID,IB,2,4,1,1,3,IE)
  IP(1)=44
  PL=35
DO 104 I=1,IK
  Y=(I,T((1+YS(I))*80/2+.5)+1)/4+2
  YY=YS(I+IK)
  IF ISTC=EQ,2,FNCODE(4,990,IB(K))YY;59 T9 104
  IF IC2(I,IPV),YE,1,59 T9 2104
  C9 1104 I=1,24

```



```

1104 I=I+4R
1104 C=C/2+2;FNCODE(8,991,IR(K))YY
1104 IF KL=40, KL=2
104 IF 'FD(I,IEV)'-EO.C, KL=KL+1;CALL TEXT0(ID,IB,24,KL,1,1,3,IE)
1104 IF 'FD(C,2,3)'-T3 108
1104 I=1,48
105 I=I+4R
105 FNCODE(4,904,I,(25))
1104 I=1,148R
106 FNCODE(8,975,IR(I*2+24))VAG9(I)
106 CALL TEXT0(IE,IR(25),24,2,1,1,3,IE)
106 FNCODE(4,982,I)
1104 I=1,148R T3 108
1107 I=1,148R
107 FNCODE(8,975,I(I*2))Y3(I)
107 CALL TEXT0(ID,IB,2-ID:IF+1,1,1,3,IE)
108 I=I+4R(C,1)
108 I=I+4R(C,-1,-.9,C);IR(3)=IPACK(1,-.9,1)
108 I=I+4R(C,-1,-.9,C);IE(5)=IPACK(-1,-.9,1)
110 I=1,148R I=5,7
109 CALL G-424(10,IR,5,1,IF)
109 I=I+4R(7777777)
110 CALL TEXT0(IE,IB,1,40,1,1,1,IE)
110 IF 'FD( T(40),2)'-E3.0, C T3 110
109 I=I+4R(T,14)
109 I=I+4R(E,3)
109 I=I+4R(1,982,I)
109 I=I+4R(I,1)
109 I=I+4R(1,1,1)
109 I=I+4R(T,3)

```

GETION LAITL
 END

SUBROUTINE PPLOT

Purpose: copies the curves displayed on the screen and prepares the arrays for use with subroutines WPLOT or NONGPLOT; also controls the transfer of points to the paper recorder, through the analog computer, if this option is selected. The subroutines WPLOT is the library subroutine VPLOT and NONGPLOT is also a library program called LONGPLOT. Besides the identifiers these programs have also modified the characters used for the line printer plot.

Parameters and variables:

M,IB,LABEL,MT - as previously described

XY - buffer for data arrangement

IJ - pointer to the option selected: 1 - VPLOT; 2 - LPLLOT; 3 - RPLLOT

LBG - storage for the number of points per curve


```

SUBROUTINE PPLGT(N,XY,IB,LABEL,MT)
COMMON IU,II,INDEX,IPLGT,ILP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,RP9W,Q
COMMON LPS(3),LINE,MAC,ARRAY,RTH,ICEUNT,IMEM,PR9B
DIMENSION YV(N,5),IB(1),IT(1),JXY(8)
I2=3;I2=2;IF LPS(3).GT.LPS(2),I3=2;I2=3
TEPEAT 902, F9K I=1,I3,I2
CALL GRAPHI(ID,IF,IE)
IF IF.NE.0,PUTPUT(101),PPLGT-902,
902 DO 902 J=1,LIG(I)
CALL UPACK(IR(J+1),XY(J,4),XY(J,I),XI)
I=IX(LPS(2),LPS(3))
DO 907 J=1,3
DO 907 I=LPS(J)+1,N
907 XY(I,J)=0.
DO 903 I=5,7
CALL GRAPHI(ID,I2(3*M/2),I,IE)
IF IF.NE.0,PUTPUT(101),PPLGT-903,
CALL UPACK(IR(3*M/2+1),X,Y,KI)
IF Y.GT..9-02.Y.LT.-.9-02.Y.E3.0.,Y=.05
DO 903 J=1-4,N,3
YV(J,5)=Y
IF IU.NE.3.G6 TO 903
XY(IJ)=XY(IJ+2)=JXY(IJ+4)=JXY(IJ+6)=4
J=1;IF IU.E3.2,J=1
JXY(J)=5;JXY(J+2)=2;JXY(J+4)=3;JXY(J+6)=1
IF IU.E3.1,64 TO 904
DO 901 J=1,3
DO 901 I=1,N/2
L=-I
TEP=XY(I,J);XY(L,J)=XY(I,J)
901 XY(I,J)=TEP
CALL MXPLOT(XY,JXY,N,4,0,0,0,0,0,0,0,N/ILP,IB)
90 TO 905
904 CALL PPLGT(XY,JXY,N,4,0,0,0,0,0,0,0)
90 TO 906

```


SUBROUTINES DEMANAL, ANACOR, DIGCOR, RECOR

Purpose: to demonstrate how the correlation can be performed by point-by-point multiplication. DEMANAL interprets the selection and calls one of the other three. ANACOR executes a periodic auto-correlation using analog multiplier and integrator and works in conjunction with CONV. DIGCOR performs the correlation entirely digitally. RECOR uses the same approach to compute the partitioned correlation.

Parameters and variables:

Arguments - as in ANALYSIS

IJ - pointer to the selection. 1 - ANACOR; 2 - AUCOR; 3 - DIGCOR; 4 - RECOR


```

300 ROUTINE DEMANAL(MABFL,LABEL,YS,YR,YC,VG,IB,MT,M)
301 CALL IU,I,INDEX,IPLT,ILP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,NP9W,Q
302 RE/L NPER
303 IF ENSIPX VG(1)
304 IF T9(201,...,304) IU
305 CALL WAC92(LABEL,M,YR,YS,VG,IB)
306 CALL D10042(LABEL,M,YR,V2,V3,IS,1)
307 CALL D10043(LABEL,M,YR,YS,VG,IB,1)
308 CALL REC9R(LABEL,M,YR,YS,VG,VG(3*M),IB,MT)

```



```

SUBROUTINE ANACOR(LABEL,N,YA,YB,YC,IB)
COMMON IU,II,INDEX,IPLT,ILP,ITR,ID,C9F,SC9F,RC9F,SNR,SP9W,RP9W,Q
COMMON LFG(3),LINE,YAC,ARRAY,RTH,ICOUNT,IMEX,PR9B
REAL VPSW
DIMENSION YA(1),YB(1),YC(1),IB(1)
JA=(N-1)/YC(N+M)=0
CALL RESET(100);II=1
DO 902 II=1,JA
CALL RESET(90)
IU=0;IF II.GT.N,IU=II-N
CALL SETLIPS(12,-1);CALL COMPUTE
901 IF IU.LT.II-AYC,IU.LT.YG,TS 901
CALL SETLIPS(12,1);CALL HOLD
CALL ACK(3,YC(I))
YC9F=0;F;CALL NORMALIZE(YC,3,JA);C9F=XC9F
CALL TISPLY(Y,YB,YC,YC,N,1,IB,0);RETURN LABEL
END

```



```

SUBROUTINE DIGCOR(LABEL,M,YA,YB,YC,IE,ISP)
COMMON IU,II,INDEX,IPLST,IPL,IIR,IU,CBF,SCBF,RCBF,SNR,SPCW,RPBW,Q
COMMON LB(3),LINE,MAG,ARRAY,RTH,ICOUNT,IMEM,PR38
REAL VPOB
PIE=3.141592653589793
JA=M+M
DO 901 I=1,JA
YC(I)=0.;J=0
REPEAT 901, WHILE J.LT.I
LEI=J;K=K-J
IF L.SI.M.MK.<.LE.O.G9 TO 901
YC(I)=YC(I)+YB(L)*YA(K)
901 J=J+1
IF ISP.EQ.0, RETURN LABEL
XCBF=XCBF+CALL NORMALIZE(YC,3,JA);CBF=XCBF
CALL TISPLAY(YA,Y3,YC,YC,M,1,I2,0);RETURN LABEL
END

```



```

SUBROUTINE PFCOR(LABEL,N,YA,YB,YC,YD,IB,MT)
COMMON IO,II,INDEX,IPL01,IPL,IIR,ID,C9F,SC9F,RC9F,SNR,SP0W,RP0W,0
COMMON LOG(3),LINE,NAC,ARRAY,RTH,ICOUNT,INEM,PR0B
REAL X00.
DIMENSION YA(1),YB(1),YC(1),YD(1),IB(1),MT(1)
IK=JA+1;JC=JA+Y;IDET=C;PROBA=PROJ=0.;IVEZ=1;JK=0
IME=7
CALL SOTLINES(14,-1)
CALL DIGCOR(901S,N,YA,YA,YC,IB,0)
TEMP=0.
901 DO 902 I=1,JA
902 TEMP=MAX(YD(I),TEMP)
YC=+.3/TEMP
TMR=1E+1000.*TEMP/Q;RTH=TMR*XC9F
903 DO 904 I=1,JC
904 YC(I)=L(I)
905 DO 906 I=1,JA
906 YD(I)=.
907 DO 908 J=1,I
908 YD(I)=YD(I)+YA(J)*YC(J)*XC9F
IF YD(I).GE.RTH,IDET=1;IM=1
909 DO 906 J=1,JC-1
909 YC(J)=YC(J+1)
IF I'.G.1,IM=C;PROBA=TEMPA+1
CY=YA(C4);Y=YA(JA-1);YA(JA)=RTH;YA(JA-1)=.3333
IC IVEZ,F0.1,IVEZ=2;CALL TISPLAY(YA,YB(M+1),YD,YC,M,2,IB,0)
IIR=INDEX;DEX=3;CALL YEGSAMP(M,YC,YA);INDEX=IND
IF IVEZ,C+2,CALL TISPLAY(YA,YC,YD,YC,M,2,IB,1)
JK=0.+2
909 YD=PA+A/JK/2
909 DO 908 I=1,24
909 IF(I)=77777777
CALL TFXPO(ID,IB,1,28,1,1,1,IE)
IF IE'.E.0,OUTPUT(101)IE,'RECOR 908 AP'
IF IDET.EC.C,ENCODE(24,991,IB)

```



```

IF ICET.EQ.1,ENC9DE(44,992,18):ENC9DE(8,993,18(12))PR08
C/E C901(12,994,18(14)):ENC9DE(8,993,18(17))THR
C/E C901(8,995,18(19)):ENC9DE(8,993,18(21))SNR
CALL TEXT9(13,18,23,28,5,1,3,1E)
IF IE.NE.0,OUTPUT(101)IF, REC9R-908,
IF ICST(14).EQ.1,99 I9991
I991=0
C/I IC 904
901 IF IE(,995)PR08,THR,JR,SNR
YA(JA)=SY:YA(JA-1)=RY
CALL TEXT9(13,18,23,28,5,1,3,1E)
IF IE.EQ.0,OUTPUT(101)IE, REC9R-801,
CALL SPLITLIS(14,1)
902 IF IC(41,3).EQ.0,99 I9 802
C/I 803 I=1,24
903 I(I)=77777777
CALL TEXT9(13,18,23,28,5,1,3,1E)
IF IE.EQ.0,OUTPUT(101)IE, REC9R-803,
RETURN LABEL
904 IF IE(,995)PR08,THR,JR,SNR
905 IF IE(,995)PR08,THR,JR,SNR
906 IF IE(,995)PR08,THR,JR,SNR
907 IF IE(,995)PR08,THR,JR,SNR
908 IF IE(,995)PR08,THR,JR,SNR
909 IF IE(,995)PR08,THR,JR,SNR
910 IF IE(,995)PR08,THR,JR,SNR
911 IF IE(,995)PR08,THR,JR,SNR
912 IF IE(,995)PR08,THR,JR,SNR
913 IF IE(,995)PR08,THR,JR,SNR
914 IF IE(,995)PR08,THR,JR,SNR
915 IF IE(,995)PR08,THR,JR,SNR
916 IF IE(,995)PR08,THR,JR,SNR
917 IF IE(,995)PR08,THR,JR,SNR
918 IF IE(,995)PR08,THR,JR,SNR
919 IF IE(,995)PR08,THR,JR,SNR
920 IF IE(,995)PR08,THR,JR,SNR
921 IF IE(,995)PR08,THR,JR,SNR
922 IF IE(,995)PR08,THR,JR,SNR
923 IF IE(,995)PR08,THR,JR,SNR
924 IF IE(,995)PR08,THR,JR,SNR
925 IF IE(,995)PR08,THR,JR,SNR
926 IF IE(,995)PR08,THR,JR,SNR
927 IF IE(,995)PR08,THR,JR,SNR
928 IF IE(,995)PR08,THR,JR,SNR
929 IF IE(,995)PR08,THR,JR,SNR
930 IF IE(,995)PR08,THR,JR,SNR
931 IF IE(,995)PR08,THR,JR,SNR
932 IF IE(,995)PR08,THR,JR,SNR
933 IF IE(,995)PR08,THR,JR,SNR
934 IF IE(,995)PR08,THR,JR,SNR
935 IF IE(,995)PR08,THR,JR,SNR
936 IF IE(,995)PR08,THR,JR,SNR
937 IF IE(,995)PR08,THR,JR,SNR
938 IF IE(,995)PR08,THR,JR,SNR
939 IF IE(,995)PR08,THR,JR,SNR
940 IF IE(,995)PR08,THR,JR,SNR
941 IF IE(,995)PR08,THR,JR,SNR
942 IF IE(,995)PR08,THR,JR,SNR
943 IF IE(,995)PR08,THR,JR,SNR
944 IF IE(,995)PR08,THR,JR,SNR
945 IF IE(,995)PR08,THR,JR,SNR
946 IF IE(,995)PR08,THR,JR,SNR
947 IF IE(,995)PR08,THR,JR,SNR
948 IF IE(,995)PR08,THR,JR,SNR
949 IF IE(,995)PR08,THR,JR,SNR
950 IF IE(,995)PR08,THR,JR,SNR
951 IF IE(,995)PR08,THR,JR,SNR
952 IF IE(,995)PR08,THR,JR,SNR
953 IF IE(,995)PR08,THR,JR,SNR
954 IF IE(,995)PR08,THR,JR,SNR
955 IF IE(,995)PR08,THR,JR,SNR
956 IF IE(,995)PR08,THR,JR,SNR
957 IF IE(,995)PR08,THR,JR,SNR
958 IF IE(,995)PR08,THR,JR,SNR
959 IF IE(,995)PR08,THR,JR,SNR
960 IF IE(,995)PR08,THR,JR,SNR
961 IF IE(,995)PR08,THR,JR,SNR
962 IF IE(,995)PR08,THR,JR,SNR
963 IF IE(,995)PR08,THR,JR,SNR
964 IF IE(,995)PR08,THR,JR,SNR
965 IF IE(,995)PR08,THR,JR,SNR
966 IF IE(,995)PR08,THR,JR,SNR
967 IF IE(,995)PR08,THR,JR,SNR
968 IF IE(,995)PR08,THR,JR,SNR
969 IF IE(,995)PR08,THR,JR,SNR
970 IF IE(,995)PR08,THR,JR,SNR
971 IF IE(,995)PR08,THR,JR,SNR
972 IF IE(,995)PR08,THR,JR,SNR
973 IF IE(,995)PR08,THR,JR,SNR
974 IF IE(,995)PR08,THR,JR,SNR
975 IF IE(,995)PR08,THR,JR,SNR
976 IF IE(,995)PR08,THR,JR,SNR
977 IF IE(,995)PR08,THR,JR,SNR
978 IF IE(,995)PR08,THR,JR,SNR
979 IF IE(,995)PR08,THR,JR,SNR
980 IF IE(,995)PR08,THR,JR,SNR
981 IF IE(,995)PR08,THR,JR,SNR
982 IF IE(,995)PR08,THR,JR,SNR
983 IF IE(,995)PR08,THR,JR,SNR
984 IF IE(,995)PR08,THR,JR,SNR
985 IF IE(,995)PR08,THR,JR,SNR
986 IF IE(,995)PR08,THR,JR,SNR
987 IF IE(,995)PR08,THR,JR,SNR
988 IF IE(,995)PR08,THR,JR,SNR
989 IF IE(,995)PR08,THR,JR,SNR
990 IF IE(,995)PR08,THR,JR,SNR
991 IF IE(,995)PR08,THR,JR,SNR
992 IF IE(,995)PR08,THR,JR,SNR
993 IF IE(,995)PR08,THR,JR,SNR
994 IF IE(,995)PR08,THR,JR,SNR
995 IF IE(,995)PR08,THR,JR,SNR
996 IF IE(,995)PR08,THR,JR,SNR
997 IF IE(,995)PR08,THR,JR,SNR
998 IF IE(,995)PR08,THR,JR,SNR
999 IF IE(,995)PR08,THR,JR,SNR
1000 IF IE(,995)PR08,THR,JR,SNR

```


REFERENCES

1. Aleksandrov, I. A., "Physical Nature of the 'Rotation Noise' of Ship Propellers in the Preserve of Cavitation," Soviet Physics-Acoustics, v. 8, pp. 23-28, July-September 1962.
2. Berkowitz, R. S., Modern Radar, John Wiley, 1965.
3. Cooley, J. W., Lewis, P., and Welch, P., "Application of the Fast Fourier Transform to Computation of Fourier Integrals," IEEE Transactions on Audio, v. 15, pp. 79-84, June 1967.
4. Gerlach, A. A., Theory and Applications of Statistical Wave Period Processor, Cook Electric, 1967.
5. Gold, B., and Jones, C. M., Digital Processing of Signals, Lincoln Laboratory Publications, 1969.
6. Horton, C. W., Signal Processing of Underwater Acoustic Waves, U. S. Government Printing Office, 1966.
7. Korn, G. A., Random-Process Simulation and Measurements, McGraw-Hill, 1966.
8. Panter, P. F., Modulation, Noise, and Spectral Analysis, McGraw-Hill, 1965.
9. Urick, R. J., Principles of Underwater Sound for Engineers, McGraw-Hill, 1967.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Associate Professor G. A. Rahe Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4. CDR Mauro Cesar R. Pereira, Brazilian Navy Rua Desembargador Isidro 10 apt C-01 Rio de Janeiro, 20000, Guanabara - BRASIL	1
5. Gabinete do MINISTRO DA MARINHA Ministerio da Marinha Brasilia, D.F., Brasil	1
6. Diretoria de Comunicacoes e Eletronica da Marinha Ministerio da Marinha Rio de Janeiro, 20000, Guanabara, BRASIL	2
7. Instituto de Pesquisas da Marinha Ministerio da Marinha Rio de Janeiro, 20000, Guanabara, BRASIL	1
8. LT Jose M. P. Saldanha Rua Arco Carvalho 31, 3 ^o apto Lisboa, Portugal	1
9. LT Antonio P. D. Souto 261 Larkin Street Monterey, California 93940	1
10. Associate Professor George L. Sackman Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	5

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Hybrid Sonar Simulation			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates) Engineer's Thesis; September 1971			
5. AUTHOR(S) (First name, middle initial, last name) Mauro Cesar Rodrigues Pereira			
6. REPORT DATE September 1971		7a. TOTAL NO. OF PAGES 188	7b. NO. OF REFS 9
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940	
13. ABSTRACT The purpose of this study was the development of a computer system for research and instruction in signal processing for underwater sound applications. The nature of the problem demanded the use of three computers interfaced by hardware and software to operate as a single system. A general purpose analog computer was employed to generate signals and noise, provide analog processing and filtering and to allow the input of actual signals collected in the field. A general purpose digital computer provided for the examination of existing processing methods and for the investigation of new techniques. A high level graphics computer was used for data display and as the system control station which provided for a high degree of human operator processing and interaction. All the hardware required was provided in the Computer Laboratory of the Naval Postgraduate School, Department of Electrical Engineering and this study prepared the software for its convenient and efficient use for this application.			

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Hybrid Sonar Simulation Signal Processing Graphics Display						

7 FEB 73
R NOV 73
12 JAN 76

BINDERY
20705
22506
22613

Thesis
P3348
c.1

Pereira

Hybrid sonar simulation.

133324

7 FEB 73
R NOV 73
12 JAN 76

BINDERY
20705
22506
22613

Thesis
P3348
c.1

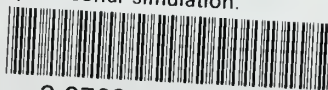
Pereira

Hybrid sonar simulation.

133324

thesP3348

Hybrid sonar simulation.



3 2768 001 00221 5
DUDLEY KNOX LIBRARY